A GROWING HIERARCHICAL SELF-ORGANIZING MAP WITH MINING ASSOCIATION RULES FOR SOFTWARE REPOSITORY ORGANIZATION AND VISUALIZATION

By

SONGSRI TANGSRIPAIROJ

Bachelor of Science (Computer Science)
Thammasat University
Bangkok, Thailand
1994

Master of Science (Computer Science)
Mahidol University
Bangkok, Thailand
1996

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December 2004

A GROWING HIERARCHICAL SELF-ORGANIZING MAP

WITH MINING ASSOCIATION RULES FOR

SOFTWARE REPOSITORY ORGANIZATION

AND VISUALIZATION

Thesis Approved:

M. Samadzadeh

Thesis Adviser

J. Chandler

H. Lee

Jerrie W. Dugger

Dean of the Graduate College

PREFACE

In this research, we first investigated the feasibility of applying data mining technology to software reuse, especially to discover useful knowledge about reusable components stored in a software repository. We also introduced a taxonomy that can be used to categorize data mining applications supporting software reuse. In addition, we proposed a new approach for software repository organization and visualization, with an attempt to facilitate the process of retrieving reusable components. The underlying idea behind the approach is the combination of two effective data mining techniques, namely the growing hierarchical self-organizing map (GHSOM) and the mining association rules.

The GHSOM is applied to cluster reusable components into groups of semantically similar ones and to ease the visualization of the structure of the software repository. Mining association rules are used to discover interesting association rules that represent a number of characteristics of the software components. The potential of the proposed approach was demonstrated on five data sets of C/C++ program source code files gathered from a number of websites. The results of the GHSOM were compared with the ones obtained by using the traditional SOM with respect to three different perspectives: visualization of the resulting maps, structure of the resulting maps, and training time. Additionally, for a particular area of the GHSOM, a number of interesting association rules were discovered and examined.

We believe that data mining technology is a feasible approach for supporting software reuse. The discovered knowledge can help developers to acquire reusable components, organize software repositories, understand the selected components, and find the most suitable components to reuse. According to the experimental results, we found that the resulting maps of GHSOM, serving as retrieval interfaces, can help developers to obtain better insight into the structure of a software repository and increase their understanding of the semantic relationships among software components. By using the resulting maps, developers can find the needed software components more easily and quickly. The GHSOM is more promising than the traditional SOM owing to its adaptive architecture and the ability to expose the hierarchical structure of data. Moreover, the interesting association rules discovered can be useful in identifying a cohesive set of include files that occur frequently together in a collection of software components.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my research advisor, Dr. M. H. Samadzadeh for his constructive guidance, valuable instruction, continuous motivation, and great dedication throughout this study. My sincere gratitude is also conveyed to my other committee members Drs. John P. Chandler, H. K. Dai, and Cecil W. Dugger for their comments and suggestions.

Moreover, I would like to thank the Computer Science Department, Oklahoma State University for offering me teaching assistantship and providing the computing supporting resources necessary for my research. Also, I am grateful to Mr. Iker Gondra, currently a Ph.D. candidate in the Computer Science Department, for our discussions and his friendship.

Furthermore, I would like to thank the Royal Thai Government, Ministry of University Affairs, and the Department of Computer Science, Faculty of Science, Mahidol University, Bangkok, Thailand for providing me this great educational opportunity and for their munificent financial support.

Finally, I would like to give my special thanks to my family and friends for their everlasting love, understanding, support, and inspiration.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

1.1 Background

Software reuse, one of the well-known concepts in software engineering, is the use of previously constructed software components [Krueger 92] or related knowledge [Frakes and Fox 95] to develop new software systems. Software reuse has been widely recognized as a promising means for enhancing the quality of software systems as well as improving the productivity of the software development process [Zand and Samadzadeh 94] [Basili et al. 96] [Samadzadeh and Zand 99] [Ravichandran and Rothenberger 03]. By reusing existing software, developers do not have to "reinvent the wheel" or waste time/cost/effort to develop software from scratch again. Furthermore, it can be argued that the reusable components have been already tested in many different circumstances. Hence, they are expected to contain little or no errors and cause few if any problems in the new contexts.

In the past, software reuse was exercised by individual developers and within small groups on an *ad hoc* basis through the informal use of their own software components constructed during their previous projects. At the present time, software reuse is commonly applied in the software development practices of many commercial companies and governmental organizations in a systematic way [Morisio et al. 02]

[Frakes and Isoda 94]. In other words, reuse oriented software construction has become a well-defined software development process.

Since more and more new software systems are designed and developed every day, the amount of software components available is progressively increasing. It becomes difficult for developers to keep up with all the potentially reusable components created. A software repository can be built to store and organize reusable components. For efficient and effective reuse, a software repository should provide a large number of reusable components over a wide spectrum of application domains [Maarek et al. 91]. It is essential that a software repository should be well structured because "the structure of a repository is key to obtaining good retrieval results" [Henninger 94]. One of the critical success factors of software reuse is that finding reusable components must be faster and easier than constructing them from scratch [Krueger 92]. Therefore, a software repository should also provide tools to help developers to locate, compare, and retrieve candidates for potential reuse [Guo and Luqi 00].

1.2 Motivation for the Research

A variety of methods have been proposed to organize software repositories and to facilitate the process of retrieving software components. Most existing methods are "either too ineffective to be useful or too intractable to be usable" [Mili et al. 98], and seem to work properly only for a small software repository. However, a realistic software repository is possibly quite large and rapidly changing. If a software repository is not well-organized, developers can be confronted with search results that are irrelevant to their stated interest. They may need to spend too much time browsing through the results

in order to select the closest match to reuse. Thus, it certainly behooves us to consider utilizing the data mining technology to assist us in organizing a large software repository in a way that can help developers to find the desired software components quickly and easily, and hence to make better decision in selecting the right components for reuse.

Data mining is a relatively new but quite advanced data analysis technique whose primary goal is to extract likely useful knowledge or hidden patterns from large and complex databases [Chen et al. 96] [Fayyad et al. 96]. This knowledge has been shown to be meaningful and useful for analysts in improving decision making, making predictions, and planning [Mitchell 99]. During the past decade, data mining has proven to be quite practical considering its implementation in a broad range of applications from business areas to scientific domains such as credit card fraud detection, credit approval, customer purchase behavior analysis, stock market prediction, medical diagnosis, protein structure discovery, aircraft components failure detection, and sky object classification [Langley and Simon 95] [Brachman et al. 96] [Fayaad et al. 96].

An analogy is drawn between mining for useful knowledge in a database and searching for reusable components in a software repository. This observation suggests that data mining tools, techniques, and approaches can be utilized to obtain interesting knowledge about software components in a software repository. This knowledge can be beneficial to software developers in finding and understanding the "closest" software components, i.e., the optimum "fits", for their needs.

The self-organizing map (SOM), which is basically an unsupervised learning neural network, is a powerful data mining technique for clustering and visualization of huge and high-dimensional data sets [Kohonen 01]. For clustering, SOM can distinguish

input data into a number of clusters by using some similarity measures. For visualization, SOM can reduce a high-dimensional input space to a two-dimensional map, which can help users to visualize structures in the original data as well as the semantic relationships among them.

Recently, a wide variety of SOM applications have been developed and reported in the literature. For example, SOM techniques have been applied to organize massive collections of documents [Kohonen et al. 00] [Merkl and Rauber 00], to analyze financial data [Deboeck and Kohonen 98], to visualize user behavior of computer systems for anomaly detection [Hoglund et al. 00], to cluster climate data [Reljin et al. 02], and to categorize DNA sequences [Naenna et al. 03].

Previous related works have utilized SOM to organize a software repository [Merkl et al. 94] [Ye and Lo 01] and to analyze software measures in order to identify the key characteristics of software systems [Pedrycz et al. 01]. One of the main reasons for the limited success of these previous related works is the use of the traditional SOM, which uses a fixed network architecture and is not able to show hierarchical relations among the input data [Rauber et al. 02]. This may result in a significant limitation on the final mapping and may not be feasible when the number of software components stored in the software repository is not known exactly.

Mining association rules is a useful data mining technique for discovering a set of important association rules in a large database based on statistical significance [Agrawal and Srikant 94]. These association rules show relationships among items, e.g., the relationship that the presence of some items in a transaction implies the presence of other items in the same transaction [Chen et al. 96].

One of the remarkable mining association rules applications is market-basket analysis, which is the process of determining which products a customer frequently buys at the same point in time or over a period of time [Agrawal et al. 93]. It allows businesses to understand customers' purchase behavior in order to gain a competitive advantage. With such valuable information, businesses can improve the quality of decisions on the placement of products in a store or the layout of mail-order catalog pages and Web pages [Brachman et al. 96] [Ganti et al. 99].

1.3 Research Objectives

The main objectives of this research are:

1) To investigate the feasibility of applying data mining technology to software reuse, particularly to discover useful knowledge from a software repository.
2) To introduce a taxonomy that can be used to categorize data mining applications supporting software reuse.
3) To propose a new approach for software repository organization and visualization, with an attempt to make a software repository well-structured and to facilitate the process of retrieving software components.
4) To demonstrate the potential of the GHSOM for the organization and visualization of a collection of reusable components stored in a software repository, and compare the results with the ones obtained by using the traditional SOM.
5) To show the usefulness of the mining association rules for the discovery of some interesting characteristics about reusable components, which are mapped onto a particular area of the GHSOM.

1.4 Organization of the Dissertation

The remainder of this dissertation is organized as follows. Chapter II provides a literature review on software reuse, data mining technology, and previous research on applying data mining technology to software reuse. Also, this chapter introduces a taxonomy of data mining applications supporting software reuse. Chapter II ends with a summary discussion of the concepts and methodology of SOM. Chapter III explains the design and methodology of the proposed approach. In this chapter, system architecture of the approach is illustrated and its four major modules, i.e., feature extraction, GHSOM construction, mining association rules, and visualization and retrieval, are explained. Chapter IV describes the experiments and the results obtained, including the experiment objectives, the data sets, the software tools and computer systems used, and the results. Chapter V gives the summary and conclusions, as well as some directions for future work. Finally, there are four appendices: Appendix A provides a glossary, Appendix B contains the detailed lists of the C/C++ program source code files in the data sets used in the experimentation, Appendix C gives some sample pages of a feature vector file, and Appendix D gives an example of a source code itemsets file.

CHAPTER II

LITERATURE REVIEW

2.1 Software Reuse

Software reuse is simply defined as "the process of creating software systems from existing software rather than building software systems from scratch" [Krueger 92]. The most outstanding benefits of software reuse over conventional software development are improving software quality and productivity, achieving savings in terms of cost, time, and effort to implement new software systems, as well as reducing maintenance costs [Zand and Samadzadeh 94] [Basili et al. 96] [Samadzadeh and Zand 99] [Ravichandran and Rothenberger 03]. However, in practice, software reuse involves some obstacles such as the NIH (not-invented-here) factor which describes the situation where developers prefer to use their own software components than the ones developed somewhere else, and inadequate tools to assist developers in representing, storing, and retrieving reusable components [Zand and Samadzadeh 94].

There are many different forms of software reuse: opportunistic or systematic, vertical or horizontal, compositional or generative, and black-box or white-box [Prieto-Diaz 93]. Opportunistic reuse is performed in an ad-hoc fashion during software development, whereas systematic reuse is planned and integrated into a well-defined

software development process. Vertical reuse is the reuse of software components within a single domain, but horizontal reuse is the reuse of software components across different domains. Compositional reuse is the use of existing software components as building blocks for new software systems, while generative reuse is reuse at the specification level using application or code generators. Black-box reuse is the reuse of software components without modification, on the other hand, white-box reuse is the reuse of software components with modification.

2.1.1 Software Repositories

The key ingredient for instituting and popularizing software reuse is the establishment of a quality software repository. By quality, we mean that the software repository should provide an adequate number of software components over a wide variety of application domains and it should be organized in such a way that developers can quickly find the desired reusable components [Maarek et al. 91]. In addition, a repository should provide tools for developers to find and understand the most suitable software components for the task at hand and support system composition and rapid prototyping [Guo and Luqi 00].

Building a software repository involves three major tasks: defining types of reusable components, defining classification methods for describing software components, and defining search and retrieval mechanisms for software developers to locate candidate components for potential reuse. These three tasks are discussed in the following three subsections.

2.1.1.1 Types of Reusable Components

Software components, also known as software assets or software artifacts, are the objects of reusability. The types of software components that can be reused are not confined to fragments of source code. A reusable component can be "any information which a developer may need in the process of creating software" [Freeman 87]. In other words, products of the software development life cycle are all candidates for reuse [Prieto-Diaz 93] [Zand and Samadzadeh 94] [Samadzadeh and Zand 99]. The following are examples of software components categorized according to the phases of the software development life cycle in which they are produced [Sommerville 04].

- *Requirement analysis and specification*: feasibility study documents, requirement documents, specification documents, etc.
- *System and software design*: design cases, design templates, design patterns, application frameworks, software architectures, user interface designs, etc.
- *Implementation and unit testing*: program/subprogram code fragments, library functions, object classes, macros, third-party software packages, etc.
- *Integration and system testing*: test plans/cases/reports, etc.
- *Operation and maintenance*: programmer's guide, user's manual, etc.

2.1.1.2 Software Classification Methods

A classification method is a way of defining a representation or description of the software components stored in a software repository [Frakes and Pole 94]. By using a classification method, the software components are systematically organized into meaningful structures that enable developers to understand software components they need without frustration and delay.

There are four primary classification methods that most existing software repositories use: enumerated classification, faceted classification [Prieto-Diaz 91], attribute-value classification, and free text keyword classification [Frakes and Pole 94]. In *enumerated classification*, a subject area is broken into a predefined hierarchical listing of all possible categories. A software component is then assigned to one of these predefined categories. Due to the well-defined hierarchy structure, it is easy for developers to understand the relationships among the indexing terms and to find reusable components by browsing up and down the hierarchy structure. However, this method requires a complete analysis of the subject area to generate hierarchical categories, and thus makes it difficult to change. In *faceted classification*, a software component is represented by a set of facets and facet values called terms. Developers can search for reusable component by specifying the most appropriate term for each facet. This method is more flexible than the enumerated classification because one facet can be changed without affecting others in the method. In *attribute-value classification*, a software component is described by a set of attributes and their values. Similar to the faceted method, attributes are equivalent to facets and values are equivalent to facet terms. But, this method does not place restrictions on the ordering of attributes and values or the number of attributes used to describe a domain. In *free text keyword classification*, a software component is associated with a number of terms that are automatically extracted from software documentation (such as manual pages and code comments) by using classic information retrieval techniques. The advantages of this method include the absence of a need for manual indexing and no restriction on the terms used to describe a software component.

In addition to these methods, a number of other classification methods have been proposed, e.g., a combination of some classification techniques [Poulin and Yglesias 93], a hierarchical thesaurus [Liao et al. 97], a multi-tiered classification scheme [Smith et al. 98], and a Reuse Description Formalism [Houhamdi and Ghoul 01].

2.1.1.3 Search and Retrieval Mechanisms

A search and retrieval mechanism is a means for developers to locate candidates for potential reuse. Two classic search and retrieval mechanisms are browsing and keyword searching [Mili et al. 99]. *Browsing* provides a natural search method for exploring a software repository. Developers can understand the relationships among indexing terms and find the desired software components by moving up and down the hierarchy structure. *Keyword searching* enables developers to confine their attention to a specific group of software components by formulating a query to express their domain of interest. A query usually consists of a set of keywords and operators (e.g., AND, OR, NOT, or double quotes). These operators are used to create complex queries and assist in query refinements.

A large number of search and retrieval mechanisms have been developed. The following are but a few examples: a generalized behavior-based retrieval [Hall 93], an incremental query refinement [Henninger 94], profile/signature matching approaches [Luqi and Guo 99], retrieval with different levels of accuracy (i.e., exact match, match, and similar) [El-Khouly et al. 99], and using a learning agent to assist the browsing of software libraries [Drummond et al. 00].

2.1.2 The Process of Reuse-Based Software Development

In general, the process of reuse-based software development (as depicted in Figure 2.1) consists of six major phases: acquisition, classification, retrieval, understanding, adaptation, and integration [Constantopoulos et al. 95].

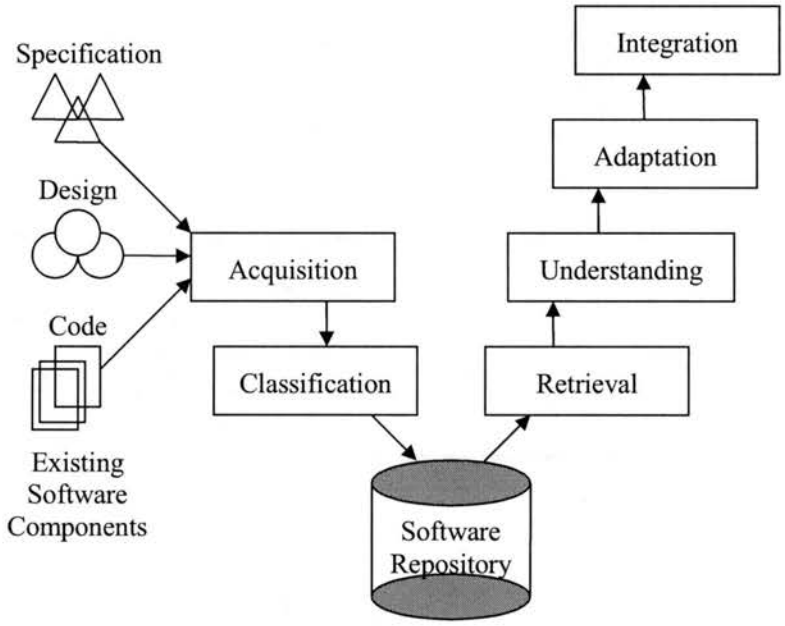


Figure 2.1 The process of reuse-based software development

- *Acquisition* is the phase in which existing software components with potential for reuse are acquired from diverse sources, e.g., in-house developers, software houses, or public domains on the Internet.
- *Classification* is the phase in which the classification system abstracts, organizes, and catalogs the acquired components according to a software classification method, e.g., faceted classification or free text keyword classification. Then, the software components and their attribute information are stored in the software repository.

- *Retrieval* is the phase in which developers look for reusable components of interest. They pass their requirements to the retrieval system by using search and retrieval tools, e.g., by browsing or by using keyword-based searching. The retrieval system performs searches on the software repository for the software components closest to the developer's needs, and then forwards the results back to the developer.
- *Understanding* is the phase in which developers study the resulting components in order to know their functionalities, structures, and methods to be able to reuse them in new applications.
- *Adaptation* is the phase in which some selected components may need to be modified to meet the new project's requirements specification.
- *Integration* is the phase in which the adapted components are integrated into new applications.

2.2 Data Mining

For decades, advanced technology in data storage, database management systems, and data warehousing has enabled organizations to accumulate a great deal of data in very large and complex databases. Unfortunately, traditional data analysis mechanisms, e.g., statistical analysis or querying systems, offer only informative summary reports, but cannot in general help extract useful knowledge. Moreover, as the quantity of data grow progressively, these mechanisms are expensive and time-consuming to exploit. Hence, data mining technology has emerged to alleviate some of this difficulty [Fayyad et al. 96].

Data mining is a relatively new and advanced data analysis technique whose primary function is to extract potentially useful knowledge from large databases [Chen et al. 96] [Fayyad et al. 96]. In this context, useful knowledge encompasses hidden patterns, possibly unknown relationships among data, trends or behaviors, and a summarization or generalization of the original data. This extracted information can be meaningful for analysts not only to understand data more deeply but also to make the process of decision making, formulating predictions, and planning more effective [Mitchell 99]. The field of data mining draws from many research fields including statistics, machine learning, artificial intelligence, database systems, knowledge-base systems, knowledge acquisition, pattern recognition, data visualization, and high performance computing [Chen et al. 96] [Fayyad et al. 96].

2.2.1 Knowledge Discovery in Databases

Data mining is also known as Knowledge Discovery in Databases (KDD). In fact, strictly speaking, data mining and KDD mean different things. KDD refers to the entire process of transforming low-level data to high level information, whereas data mining is one of the fundamental steps of the KDD process. Data mining usually constitutes approximately 15%-25% of the effort of the overall KDD process [Brachman et al. 96]. A general definition of KDD is "the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" [Fayyad et al. 96].

Once an application domain is thoroughly studied and the goals of the data mining process are defined, the KDD process begins. The KDD process typically consists of five main steps: selection, preprocessing, transformation, data mining, and

interpretation (as depicted in Figure 2.2) [Fayyad et al. 96]. *Selection* is the step that involves selecting a subset of the data used as the target data set or input for data mining. *Preprocessing* is the step that prepares the target data set for analysis and performs basic operations such as eliminating duplicate data, rectifying inconsistent data, and handling missing data fields. *Transformation* is the step that converts the data, especially the non-numerical values, into meaningful numerical values. This step is important for data mining algorithms such as neural networks and genetic algorithms that employ numerical values as their inputs. *Data mining* is the step that finds hidden patterns in the transformed data by using proper data mining algorithms. *Interpretation* is the step that interprets the resulting patterns and presents them to users in an understandable way.

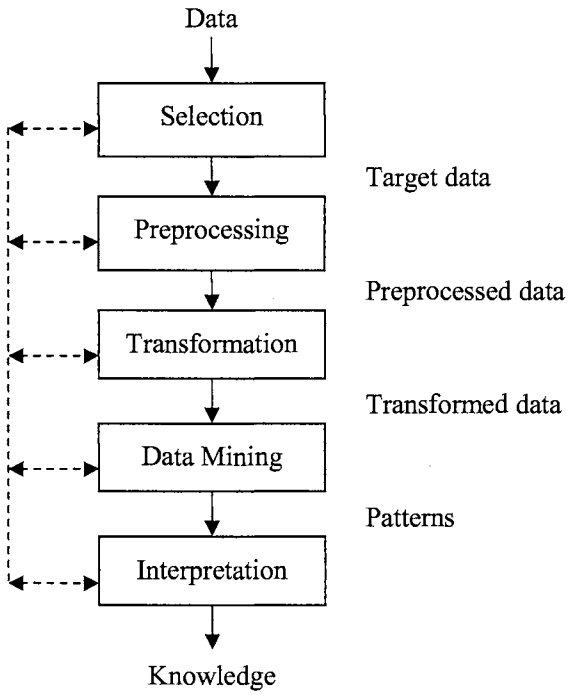


Figure 2.2 The process of knowledge discovery in databases [Fayyad et al. 96]

The KDD process is not a simple linear model, rather it is interactive and iterative in nature. The result of any one step may cause changes in the preceding or succeeding steps. For this reason, the KDD process may contain various feedback loops [Fayyad et al. 96].

2.2.2 Data Mining Tasks and Techniques

In general, data mining has two different goals: 1) prediction, to help forecast the future behavior and 2) description, to present the patterns in the data to users in a comprehensible form [Fayyad et al. 96]. Based upon these goals, data mining can carry out a wide spectrum of tasks such as classification, clustering, associations, regression, summarization and generalization, dependency modeling, change and deviation detection, model visualization, and exploratory data analysis [Chen et al. 96] [Fayyad et al. 96] [Goebel and Gruenwald 99].

To cope with the diversity of possible different types of tasks, various data mining techniques together with their attendant efficient algorithms have been developed and deployed by researchers over the last few years. Here are some examples: decision trees, mining association rules, clustering, neural networks, genetic algorithms, case-based reasoning, statistical methods, Bayesian belief networks, fuzzy sets, and rough sets [Chen et al. 96] [Fayyad et al. 96] [Goebel and Gruenwald 99]. A brief description of three of the best-known data mining tasks as well as their prominent techniques and algorithms are given below.

1) Classification

The task is to classify data items into one of several predefined classes based on

the values of certain attributes [Chen et al. 96]. For example, in massive customer databases, classification can be used to categorize customers according to their preference for magazines. A decision tree constructed from a training set of data items is the most commonly used technique for classification. A decision tree represents useful knowledge consisting of non-leaf nodes and leaf nodes, where a non-leaf node denotes a test on a single attribute value and a leaf node denotes a class. The tree is subsequently used to classify new data items, whose classes are unknown, by testing their attribute values beginning at the root node and ending at a leaf node. Well-known examples of decision tree algorithms are ID3, CART, XAID/CHAID, and C4.5 [Kleissner 98].

2) Clustering

The task is to divide data items into classes or clusters according to similarity which is quantified by a numerical measure such as the Euclidean distance, the squared Mahalanobis distance, or the Hausdorff distance [Jain et al. 99]. Unlike classification, these classes are not predefined but determined from the data. For example, clustering can be used to find subgroups of customers having similar purchase behaviors. Important clustering techniques are hierarchical clustering algorithms, partition algorithms, nearest neighbor clustering, fuzzy clustering, artificial neural networks used for clustering (e.g., Kohonen's learning vector quantization (LVQ) and self-organizing map (SOM)), and evolutionary approaches for clustering (e.g., genetic algorithms and evolution strategies) [Jain et al. 99].

3) Associations

The task is to derive a set of association rules (showing relationships or dependencies among attributes and data items) based on statistical significance [Agrawal

et al. 93]. For example, associations can be used for market-basket analysis, which is the process of determining which products a customer typically purchases at the same time. This kind of information can help retailers to understand the customers' purchase behaviors and lead to improved decisions on product location and promotion. The Apriori algorithm [Agrawal et al. 93] is the pioneering algorithm for mining association rules. A large number of successor algorithms have been proposed to enhance the performance of the Apriori algorithm, e.g., partition-based algorithm, hash-based algorithm, sampling-based algorithm, Dynamic Itemset Counting algorithm, mining generalized and multi-level association rules, and mining sequential patterns [Chen et al. 96] [Ganti et al. 99].

2.3 Existing Data Mining Applications Supporting Software Reuse

As evidenced by the published literature, there has been a lot of attention on applying data mining to support software reuse. Many researchers have come up with various ideas of using data mining techniques to discover useful knowledge about software components in a software repository. As necessary background work for this research, we explored several such applications. For each application, we examined its distinctive characteristics. These applications can be categorized into three main groups based on the data mining task: 1) classifying software components, 2) clustering software components, and 3) mining reuse patterns. These three groups are discussed in the following three subsections. The fourth subsection below provides a detailed comparison of the well-known existing applications.

2.3.1 Classifying Software Components

Esteva [Esteva 90] proposed *the Inductive Classification (IC) system* to determine whether or not a software module has potential for reusability. The IC system applies inductive learning techniques to produce a decision tree to classify modules into two classes: reusable and non-reusable. Each module is described by a set of attributes that are measured in terms of software complexity metrics associated with various aspects of program structure including modularity, cohesion, coupling, size, data structure, control structure, and documentation. A sample of 81 Pascal programs was used as experimental data. Figure 2.3 shows an example of a decision tree produced by the IC system [Esteva 90].

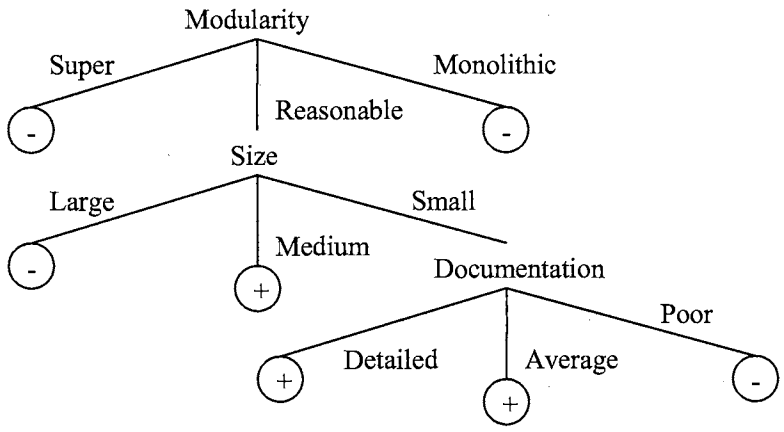


Figure 2.3 An example of a decision tree produced by the IC system [Esteva 90]

Damiani and Fugini [Damiani and Fugini 96] proposed a fuzzy classification model for a software repository containing the descriptors of the components. This model is implemented in *the Fuzzy Classification of Components (FCC) system.* Compared to the classic classification methods, this model is more flexible and useful when the

characteristics of the desired components are not completely defined. The component descriptors include fuzzy-weighted keyword pairs describing components functionalities extracted from an object-oriented code segment and its design documentation. The system also introduced a tuning function that observes user reactions to query answers from the system, and slowly adjusts the fuzzy weights.

Ugurel and his colleagues [Ugurel et al. 02] demonstrated a Support Vector Machine (SVM) approach in *the SVM system* to classify archived source code into eleven application topics and ten programming languages. SVM classifiers are trained on examples of a given programming language or programs in a specified category. Each program is represented by a binary feature vector which is derived from features extracted from the code, comments, and the README files. The demonstration was conducted with hundreds of source code files in different languages and application topics obtained from several archives on the Internet.

2.3.2 Clustering Software Components

Maarek and her colleagues [Maarek et al. 91] invented *the GURU system* to construct a software repository from a collection of software components. The system uses an indexing scheme based on the notions of lexical affinities and quantity of information extracted from documentation to represent a component. The system applies hierarchical agglomerative clustering methods to generate a browse hierarchy, which guides the search for appropriate software components. This technology has been applied to construct a repository of 1100 AIX utilities. Figure 2.4 shows a portion of a browse hierarchy produced by the GURU system [Maarek et al. 91].

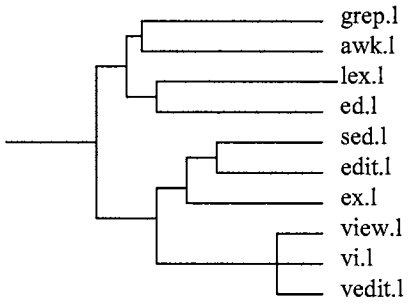


Figure 2.4 A part of a browse hierarchy produced by the GURU system [Maarek et al. 91]

Merkl and his colleagues [Merkl et al. 94] implemented *the Self-Organizing Feature Map (SOFM) system* to organize a software repository according to the semantic similarity or functional similarity of the software components. The system uses the self-organization map (SOM) technology, an unsupervised learning paradigm of neural networks, to create a two-dimensional map that helps visualize the structure of the software repository, where software components having similar behavior are mapped onto geographically closer regions of the map. Each component is represented by a feature vector, which consists of 39 features of keywords extracted from the documentation. A set of 36 MS-DOS commands is contained in the experimental repository. Figure 2.5 shows an example of a 10 ×10 map generated by the SOFM system [Merkl et al. 94].

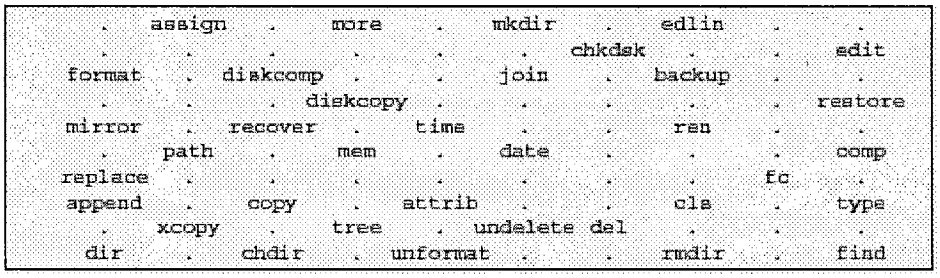


Figure 2.5 An example of a 10 × 10 map generated by the SOFM system [Merkl et al. 94]

Ye and Lo [Ye and Lo 01] also applied the SOM technique as prescribed in *the Software Self-Organizing Map (SSOM) system.* The design goals of the SSOM system were similar to those of the SOFM system (as discussed above). However, SSOM was subsequently improved to identify keywords associated with software components based on automatic indexing (weight single term, phrase, and thesaurus indexing) rather than manual indexing (binary single term indexing) as used in the SOFM system. Each software component is represented by a feature vector, which consists of 827 features or keywords extracted from the documentation. The method has been applied to a collection of 97 UNIX commands.

Pedrycz and his colleagues [Pedrycz et al. 01] developed *the SOM Clustering Analysis (SOMCA) system* by using the SOM technique in a new dimension, specifically to analyze software measure data. Each software component is characterized by a set of software metrics, e.g., lines of code, number of methods, depth of inheritance tree, and number of children. A sample of 643 JAVA classes was used as experimental data. Three different types of maps representing different aspects of the analyzed data are generated by the system: 1) weight map, which helps to identify software component profiles of clusters, 2) clustering map, which helps to distinguish clusters, and 3) data distribution map, which helps to find data popularity for each cluster.

Lee and his colleagues [Lee et al. 98] used genetic algorithms in *the Reusable Class Library (RCL) system* with the goal of finding optimized clusters into which software components are classified, and finding an optimal query which retrieves clusters containing software components similar to a given query. The system characterizes software components with the faceted classification method.

2.3.3 Mining Reuse Patterns

Michail [Michail 00] developed *the CodeWeb system* to discover reuse patterns of library classes and member functions that are normally reused in combination by application classes. The system uses "generalized association rules", which improve upon the standard association rules mining technique by taking into account the inheritance hierarchy. Each application class, serving as a component, is associated with a set of items that indicate reuse relationships involving library classes or member functions. Five reuse relations were considered: class inheritance, class instantiation, function invocation, function overriding, and implicit invocation. The demonstration was conducted with 76 C++ applications in order to mine reuse patterns for the KDE 1.1.2 core libraries. Figure 2.6 shows an example of reuse patterns discovered by the CodeWeb system [Michail 00].

class_instantiates:kdelibs'KApplication =>	Confidence	Supporters	Detractors
1. class_calls:kdelibs'KApplication::exec()	72.3%	47	18
2. class_instantiates:kdelibs'KTopLevelWidget^	58.5%	38	27
3. class_calls:kdelibs'KApplication::setMainWidget()	53.8%	35	30
4. class_calls:kdelibs'KTopLevelWidget^::show()	46.2%	30	35
5. class_instantiates:qt'QFile	24.6%	16	49
6. class_calls:kdelibs'KTopLevelWidget^::restore()	24.6%	16	49

Figure 2.6 An example of reuse patterns discovered by the CodeWeb system [Michail 00]

2.3.4 Comparisons of Existing Applications

Tables 2.1, 2.2, and 2.3 below show comparisons of the data mining applications described above based on the following six perspectives.

1. Purpose – What is the purpose of the application?
2. Phase – Which phase in the process of reuse-based software development does the application support?
3. Data Mining Task – What is the data mining task?

4. Data Mining Technique – What data mining technique does the application use?

5. Software Component – What software components are analyzed?

6. Software Representation – How are software components represented?

Table 2.1 Comparison of applications by purpose and phase

Application	Purpose	Phase
IC	To classify software modules into two classes: reusable and non-reusable.	Acquisition
FCC	To classify software components based on fuzzy weighting.	Classification and Retrieval
SVM	To classify archived source code by application topic and by programming language.	Classification
GURU	To construct software libraries from a collection of software components.	Classification and Retrieval
SOFM	To organize a software library according to the semantic similarity of software components.	Classification and Retrieval
SSOM	To organize a software library according to the semantic similarity of software components.	Classification and Retrieval
SOMCA	To identify software module clusters and their characteristics.	Classification
RCL	To find optimized clusters and an optimal query for component retrieval.	Classification and Retrieval
CodeWeb	To discover reuse patterns of library classes and member functions that are usually reused in combination by application classes.	Understanding

Table 2.2 Comparison of applications by data mining task and technique

Application	Data Mining Task	Data Mining Technique
IC	Classification	Decision Trees
FCC	Classification	Fuzzy Techniques
SVM	Classification	Support Vector Machines (SVMs)
GURU	Clustering	Hierarchical Agglomerative Clustering
SOFM	Clustering	Neural Networks (Self-Organizing Maps)
SSOM	Clustering	Neural Networks (Self-Organizing Maps)
SOMCA	Clustering	Neural Networks (Self-Organizing Maps)
RCL	Clustering	Genetic Algorithms
CodeWeb	Assoications	Mining Generalized Association Rules

Table 2.3 Comparison of applications by software component and representation

Application	Software Component	Software Representation
IC	Pascal programs	Software complexity metrics
FCC	C++ programs and design documentation	Free text keyword
SVM	Programs in various languages and application topics	Free text keyword
GURU	AIX utilities	Free text keyword
SOFM	MS-DOS commands	Free text keyword
SSOM	UNIX commands	Free text keyword
SOMCA	Java classes	Software complexity metrics
RCL	Components generated	Facet
CodeWeb	C++ programs	Items indicating reuse relationships

From the above three tables, we can make the following observations.

- Most of the applications support the Classification and Retrieval phase with the aim of organizing a software repository in such a way that helps developers in searching for the desired software components.
- None of the applications supports Adaptation and Integration phases.
- Clustering is the most popular data mining task practiced.
- The data mining technique applied is related to the data mining task and the purpose of the application.
- The reusable components selected for the analyses, are operating system commands and software modules.
- Free text keyword is the method commonly used for software representation.

2.4 A Taxonomy of Data Mining Applications Supporting Software Reuse

A taxonomy is a classification of items in a systematic way based on their inherent properties and relationships. In addition to serving as a descriptive facility to distinguish among existing items, a taxonomy typically contains provisions for not only predicting items not among its baseline set, but also the ability to prescribe new items.

A taxonomy is proposed to categorize data mining applications supporting software reuse. The taxonomy is based on two major characteristics of the applications: data mining task and data mining technique.

- *Data Mining Task:* Possible data mining tasks are classification, clustering, associations, regression, summarization and generalization, dependency modeling, change and deviation detection, model visualization, exploratory data analysis, etc.
- *Data Mining Technique*: Possible data mining techniques are decision trees, mining association rules, clustering, neural networks, genetic algorithms, case-based reasoning, statistical methods, Bayesian belief networks, fuzzy sets, rough sets, etc.

The data mining technique to be applied is related to the data mining task. For example, decision trees are usually used for the classification task and not for the clustering task. Neural networks can be applied for both classification task (with predefined classes) and clustering task (without predefined classes). The mining association rules are used exclusively for the association task.

The taxonomy of data mining applications supporting software reuse takes the form shown in Figure 2.7. Although not exhaustive due to space limitations, we believe

that the taxonomy provides a predictive framework to help identify possible new data mining applications.

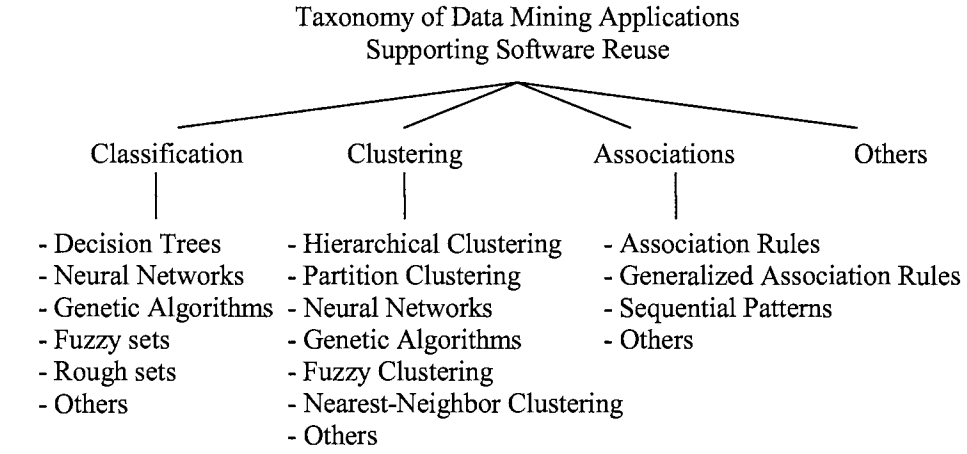


Figure 2.7 A taxonomy of data mining applications supporting software reuse

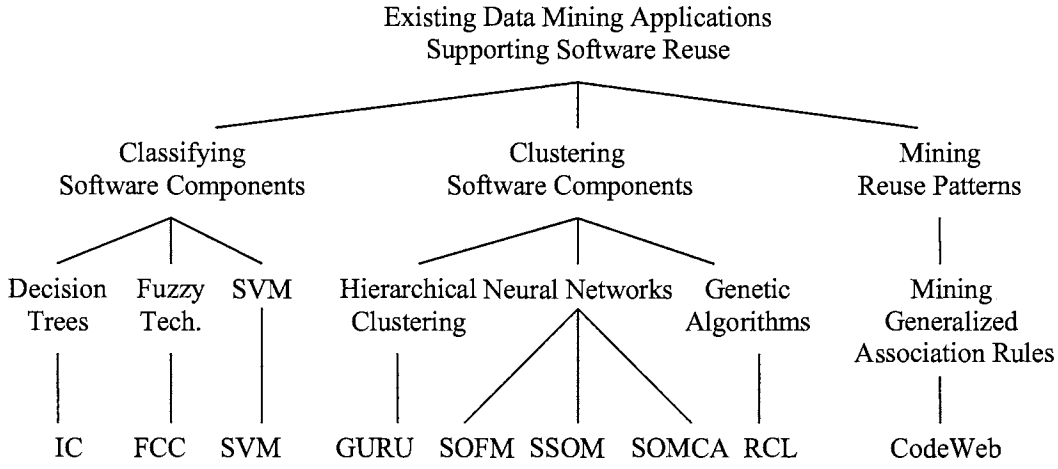


Figure 2.8 An application of the taxonomy to the existing applications

As illustrated in Figure 2.8, the taxonomy given in Figure 2.7 was applied to the existing applications mentioned previously in Section 2.3. First, the applications are categorized into three groups based on the data mining task: Group I – classifying

software components, Group II – clustering software components, and Group III – mining reuse patterns. At the second level, within Group I, there are three subgroups using decision tree technique, fuzzy technique, and support vector machine (SVM). Three subgroups using hierarchical clustering, neural networks, and genetic algorithms belong to Group II. Group III has one group using the mining generalized association rules technique. At the third level, a leaf node is labeled with the application name and indicates the class of an application. For example, the IC system is in the class of classifying software components with decision trees. The SOFM system belongs to the class of clustering software components with neural networks. The CodeWeb system is in the class of mining reuse patterns with mining generalized association rules.

2.5 Self-Organizing Map

The self-organizing map (SOM), first introduced by Kohonen in 1981 [Kohonen 01], is one of the major unsupervised learning paradigms in the family of artificial neural networks. These networks are inspired by the structure and function of the human brain, which is composed of millions of biological neurons working together. Similarly, an artificial neural network consists of a massive number of artificial neurons, which are simple and highly interconnected processing units operating in a parallel manner.

2.5.1 The Traditional Self-Organizing Map

The SOM network is typically a two-layer neural network consisting of an input layer and an output or competitive layer. The input layer is composed of a set of n-dimensional input vectors $x = [x_1, x_2, ..., x_n]^T$, where n indicates the number of features

that each input vector contains. The output layer is an m-dimensional (usually two-dimensional) grid consisting of a set of neurons, each associated with an n-dimensional weight vector $w_i = [w_{i1}, w_{i2}, ..., w_{in}]^T$ (with same dimension as the input vector). The weight vector expresses the relative importance of each input to a neuron in the grid. The arrangement of the neurons can be rectangular or hexagonal. The architecture of a 4×5 SOM is shown in Figure 2.9.

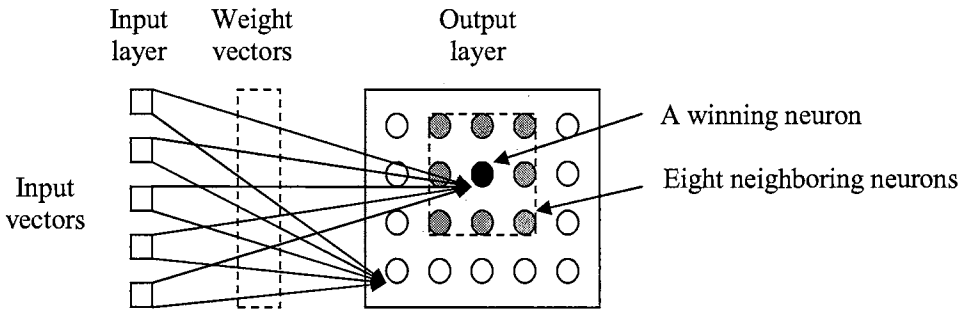


Figure 2.9 The architecture of a 4×5 SOM [Kohonen 01]

Basically, the SOM takes a set of inputs and maps them onto the neurons of a two-dimensional grid. Since SOM is an unsupervised learning algorithm, there is no target output available for the input. Hence, the SOM network learns only from its input through repetitive adjustments of the weights of the neurons. The weight vectors are randomly initialized at the first stage. Then, the SOM network performs learning in two main steps: determining a winning neuron and adjusting weights, as described below.

1) Determining a winning neuron

The SOM network determines the winning neuron for a given input vector, selected randomly from the set of all input vectors. For every neuron on the grid, its weight vector is compared with the input vector by using some similarity measure, e.g.,

Euclidean distance, Hamming distance, or Tchebyschev distance. The neuron whose weight vector is closest to the input vector in the n-dimensional space is selected to be the winning neuron. Equation (1) shows how to determine the winning neuron *c*.

$$c: \|x - w_c\| = \min_i \|x - w_i\| \qquad (1)$$

2) Adjusting weights

After a winning neuron is determined, the weight vectors of the winning neuron and all of its neighboring neurons are adjusted by moving toward the input vector according to the learning rule, as given in Equation (2)

$$w_i(t+1) = w_i(t) + h_{ci}(t)\,[x(t) - w_i(t)] \qquad (2)$$

where t is a discrete time constant denoting the current learning iteration. The neighborhood function $h_{ci}(t)$ is used to determine to which extent the neighboring neurons, lying within a certain radius of the winning neuron, will be updated. This function is a time decreasing function that converges to zero for large values of t. A typical smooth Gaussian neighborhood function is given below in Equation (3) below

$$h_{ci}(t) = \alpha(t) \exp\left(- \|r_c - r_i\|^2 / 2\sigma(t)^2\right) \qquad (3)$$

where $\alpha(t)$ is the learning rate function which controls the amount of weight vector movement and gradually decreases over time, $\sigma(t)$ is the width of the Gaussian kernel, and $\|r_c - r_i\|^2$ is the distance between the winning neuron and neuron i.

This learning process proceeds repeatedly until it converges to a stable state where there are no further changes made to the weight vectors when they are presented with the given input vectors. After the learning has been completed, an orderly map is

formed in such a way that the topology of the data is preserved and becomes geographically explicit, i.e., clusters of most similar input data appear close to one another on nearby regions of the map [Deboeck and Kohonen 98] [Kohonen 01].

2.5.2 The Dynamic Self-Organizing Map

A primary drawback of the traditional SOM is that the size of the grid and the number of neurons have to be determined in advance. This might not be feasible for some applications and result in a significant limitation on the final mapping [Blackmore and Miikkulainen 93] [Fritzke 94] [Fritzke 95] [Bauer and Villmann 97] [Alahakoon et al. 00] [Rauber et al. 02]. Several dynamic SOM models have been proposed recently to reduce the limitations of the fixed network architecture of the traditional SOM. The models rely on an adaptive architecture where neurons and connections are inserted into or removed from the map during their learning process according to the particular requirements of the input data. Some of the major variations of dynamic SOM models are summarized below.

1) Incremental Grid Growing (IGG)

The IGG algorithm [Blackmore and Miikkulainen 93] starts from an initial structure consisting of four connected neurons. Using a growth heuristic, the addition of new neurons is permitted only at the boundary of the map, expanding the map outward. In the course of the learning process, connections between neighboring neurons may be inserted or removed according to some threshold values based on the similarity of their weight vectors. This may result in several separated substructures representing distinct clusters of input data.

2) Growing Cell Structures (GCS)

The GCS algorithm [Fritzke 94] builds a network of neurons whose basic building blocks are triangles rather than two-dimensional grids of neurons. The GCS starts with a triangle of three neurons. During the learning process, some heuristic measure is used to determine where to add new neurons and which existing neurons should be deleted from the grid. The connections between nodes are adjusted in order to keep the triangular connectivity. The algorithm results in a network graph structure consisting of a set of nodes and the connections among them.

3) Growing Grid (GG)

The GG algorithm [Fritzke 95] builds a network structure of a rectangular grid. Starting with a square of four (2×2) neurons, the network grows by inserting complete rows or columns of neurons, thus always maintaining a rectangular grid structure. The areas of the grid for insertion of new neurons are determined by the computation of some heuristic measure, e.g., the winner counter for each neuron. The growth process terminates when a stopping criterion is satisfied, e.g., a desired network size is reached.

4) Growing Self-Organizing Map (GSOM)

The GSOM algorithm [Alahakoon et al. 00] is quite similar to the IGG algorithm discussed above in that an initial structure has four neurons and the new neurons are always added at the boundary of the map. The major difference with IGG is that GSOM uses a spread factor to measure and control the spread of the map. The data analyst can select regions of interest for further analysis in order to obtain a more detailed view of the clusters. A separate map is generated for each selected region. This obviously results in manually created hierarchical clusters.

5) Growing Hierarchical Self-Organizing Map (GHSOM)

The GHSOM algorithm [Rauber et al. 02] builds a hierarchical structure of multiple layers, where each layer is composed of several independent growing SOMs. At the first layer, GHSOM starts with one map consisting of a small number of initial neurons. By using the GG algorithm discussed above, the individual maps on each layer are trained independently and get specialized to a subset of the input data. When the neurons represent input data that are too diverse, they are expanded to form a new small growing SOM at a subsequent layer, where the respective data is represented in more detail. The resulting maps reflect the hierarchical structure inherent in the data.

Several researchers have explored the use of SOM combined with other machine learning or data mining techniques in order to improve the performance and obtain better results. For example, some researchers embed fuzzy sets theory into SOM [Sum and Chan 94] [Vuorimaa 94] [Chi et al. 00] [Drobics et al. 01] [Tenhangen et al. 01] and some researchers use SOM together with genetic algorithms [Tanaha et al. 96] [Ha et al. 99] [Kirk and Zurada 01] [Jin et al. 03].

CHAPTER III

DESIGN AND METHODOLOGY

3.1 System Architecture

The overall system architecture of the proposed approach is illustrated in Figure 3.1. It consists of four major modules: feature extraction, GHSOM construction, mining association rules, and visualization and retrieval. A detail description of each module is given in the next four sections.

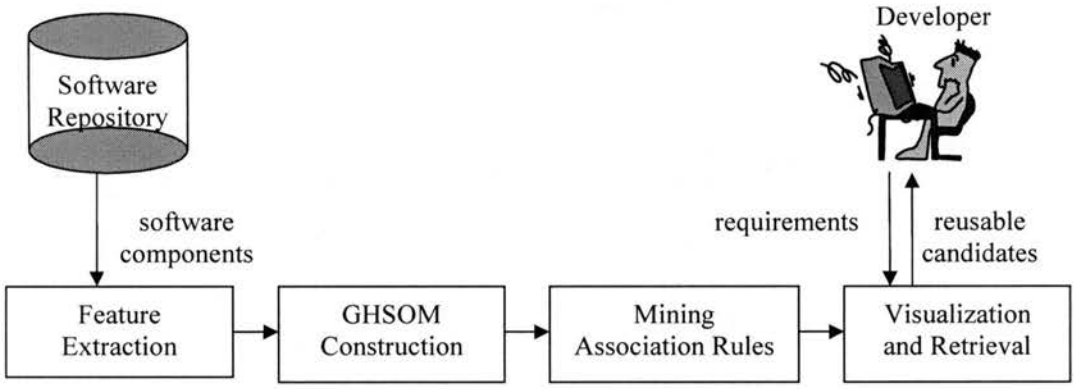


Figure 3.1 System architecture of the proposed approach

In Figure 3.1, a software repository is a place where a set of software components for potential reuse (i.e., C/C++ program source code files) was gathered and saved as text

files. The feature extraction module extracts keywords and source code items from the software components and then creates the representation for the software components including feature vectors for the GHSOM construction module and source code itemsets for the mining association rules module. Next, all of the feature vectors, served as input vectors, were fed to the GHSOM construction module to generate the resulting map. Then, focusing on a particular area of the map, the mining association rules module discovers some interesting characteristics about the software components from the source code itemsets. Finally, in the visualization and retrieval module, the resulting map, functioned as a retrieval interface, was presented to the developer.

3.2 Feature Extraction

The feature extraction module performs two important tasks: 1) creating feature vectors for the GHSOM construction module, based on the extracted keywords and 2) creating source code itemsets for the mining association rules module, based on the extracted include files.

3.2.1 Feature Vectors for the Construction of SOM and GHSOM

The procedure of creating feature vectors is shown in Figure 3.2. First, the preprocessing, including eliminating "stopwords" and stemming, was done on the

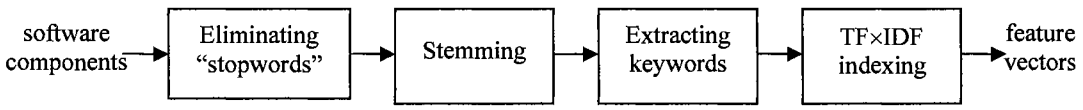


Figure 3.2 The procedure of creating feature vectors

collection of software components in order to obtain high-quality features for describing the software components. Stopwords (frequently-used words that don't help distinguish one component from the other such as "a", "an", "the", "is", "are", etc.) were eliminated. Words having the same common linguistic roots were grouped together according to the Porter Stemming algorithm [Porter 80]. For example, adjustable, adjustment, and adjust are grouped into adjust. After that, meaningful keywords were extracted from the software components by using a single-term free-text indexing scheme. During indexing, keywords occurring in less than a certain number of software components (minimum component frequency) or in more than a certain number of software components (maximum component frequency) were omitted because these keywords are too common or too rare to differentiate between different content clusters. A number of keywords are remaining for software component representation. By taking into account the importance of each keyword, these keywords are further weighted according to the term frequency multiplied by inverse document frequency (TF×IDF) weighting scheme. Each software component is then represented as a feature vector in the Vector Space Model (VSM), where the tf×idf value of each keyword in each software component is recorded in a components-versus-keywords matrix [Salton 89]. These feature vectors serve as input vectors for the construction of GHSOM.

3.2.2 Source Code Itemsets for Mining Association Rules

There are a variety of attractive items that can be extracted from program source code files. Such items include classes, member functions, variables, include files, relationships between classes (e.g., class inheritance and class instantiation), relationships

between functions (e.g., function invocation and function overriding), and metrics information (e.g., number of lines of code and comments, number of classes and methods, and the depth of the inheritance tree). In this dissertation, we are particularly interested in the include files that software components contain because they may be useful in identifying a cohesive set of include files that occur frequently together in a given set of software components.

The procedure of creating source code itemsets consists of two steps. In the first step, each software component is analyzed in order to extract all include files that it contains. In the second step, constituted from the extracted include files, an itemset is created to represent a software component. These itemsets serve as input transactions for mining association rules.

3.3 GHSOM Construction

The GHSOM Construction module builds a GHSOM for a software repository by applying the GHSOM algorithm [Dittenbach et al. 00] [Rauber et al. 02], which is an extension to the growing grid SOM [Fritzke 95] and hierarchical SOM [Miikkulainen 90]. The GHSOM can build a hierarchy of multiple layers where each layer consists of several independent growing SOMs. The size of these SOMs and the depth of the hierarchy are determined during its learning process according to the requirements of the input data.

As depicted in Figure 3.3, the architecture of a GHSOM is similar to a tree structure where the SOM(s) at each layer can branch out to additional SOMs at the

subsequent layer. The upper layers show a coarse organization of the major clusters in the data, whereas the lower layers offer a more detailed view of the data.

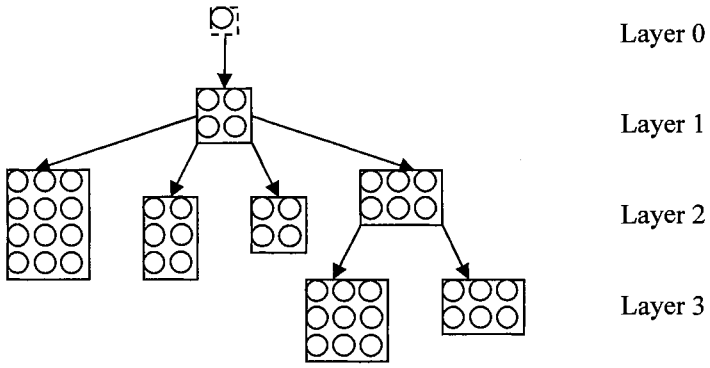


Figure 3.3 The architecture of a GHSOM [Rauber et al. 02].

For the initial setup of the GHSOM, at Layer 0, a single-neuron SOM is created and the neuron's weight vector is initialized as the average of all input vectors. Then, the learning process starts at Layer 1 with a small SOM (usually a 2×2 grid) whose weight vectors are initialized to random values.

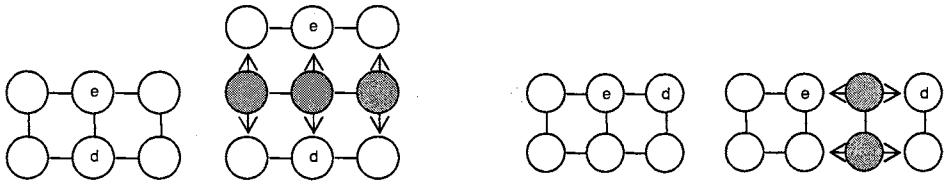
The GHSOM grows in two dimensions: horizontally (by increasing the size of each SOM) and hierarchically (by increasing the number of layers). For horizontal growth, each SOM modifies itself in a systematic way very similar to the growing grid [Fritzke 95] so that each neuron does not represent too large an input space. For hierarchical growth, the principle is to periodically check whether the lowest layer SOMs have achieved sufficient coverage for the underlying input data. The basic steps of the horizontal growth and the hierarchical growth of the GHSOM are summarized in Table 3.1 and 3.2 below.

Table 3.1 Basic steps of the horizontal growth of the GHSOM

Basic steps of horizontal growth:
1. Initialize the weight vector of each neuron with random values.
2. Perform the traditional SOM learning algorithm for a fixed number λ of times.
3. Find the error unit e and its most dissimilar neighbor unit d. (Note that the error unit e is the neuron with the largest deviation between its weight vector and the input vectors it represents.)
4. Insert a new row or a new column between e and d (See Figure 3.4). The weight vectors of these new neurons are initialized as the average of their neighbors.
5. Repeat steps 2-4 until the mean quantization error of the map $MQE_m < \tau_1 * qe_u$, where qe_u is the quantization error of the neuron u in the preceding layer of the hierarchy.

Table 3.2 Basic steps of the hierarchical growth of the GHSOM

Basic steps of hierarchical growth:
1. Check each neuron to find out if its $qe_i > \tau_2 * qe_0$, where qe_0 is the quantization error of the single neuron of Layer 0, then assign a new SOM at a subsequent layer of the hierarchy.
2. Train the SOM with input vectors mapped to this neuron.



(a) Inserting a row (b) Inserting a column

Figure 3.4 Inserting a row or a column of neurons to a SOM [Rauber et al. 02]

The growth process of the GHSOM is controlled by the following four important factors.

- The quantization error of a neuron i, qe_i, is calculated as the sum of the distance between the weight vector of neuron i and the input vectors mapped onto this neuron.
- The mean quantization error of the map (MQE_m) is the mean of all neurons' quantization errors in the map.
- The threshold τ_1 is for specifying the desired level of detail that is to be shown in a particular SOM.
- The threshold τ_2 is for specifying the desired quality of input data representation at the end of the learning process.

3.4 Mining Association Rules

Mining association rules is an efficient data mining technique for discovering a set of important association rules in a large database based on statistical significance [Agrawal and Srikant 94]. These association rules show relationships among items, e.g., the relationship that the presence of some items in a transaction implies the presence of other items in the same transaction [Chen et al. 96].

Formally, association rules are defined as follows [Agrawal and Srikant 94]: Let $I = \{i_1, i_2, ..., i_m\}$ be a set of items. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. Let X be a set of items such that $X \subseteq I$. A transaction T is said to contain X if and only if $X \subseteq T$. An association rule is an implication of the form "$X \Rightarrow Y$ *[c, s]*", which means X predicts Y with *confidence c* and *support s*. X is the antecedent of the rule and Y is the consequent of the rule, where $X \subset I$, $Y \subset I$, and $X \cap Y$

$= \varnothing$. The rule $X \Rightarrow Y$ holds in the transaction set D with *confidence* c if $c\%$ of the transactions in D that contain X also contain Y. The rule $X \Rightarrow Y$ has *support* s in the transaction set D if $s\%$ of the transactions in D contain $X \cup Y$. Confidence is a measure of a rule's strength whereas a support indicates its statistical significance [Agrawal et al. 93] [Agrawal and Srikant 94].

For example, given a database of sales transactions, suppose that in 90% of the transactions in which customers purchase bread and butter, they also purchase milk. Additionally, 5% of the transactions include all three items: bread, butter, and milk. In that case, the corresponding association rule is "bread $\wedge$ butter $\Rightarrow$ milk [90%, 5%]" with confidence 90% and support 5%. The antecedent of rule X consists of bread and butter and the consequent of rule Y consists of milk alone [Agrawal et al. 93].

The problem of mining association rules can be decomposed into two subproblems: (1) find the large item sets, i.e., the sets of items that have support above a predetermined minimum amount, and (2) use the large item sets to generate all association rules whose confidence is above a predetermined minimum amount [Agrawal and Srikant 94] [Chen et al. 96].

A number of algorithms for finding association rules have been presented in the literature, with the Apriori algorithm being the pioneering one [Agrawal and Srikant 94]. Most of the popular algorithms are variations and improvements on the Apriori algorithm, for instance the partition-based algorithm, hash-based algorithm, sampling-based algorithm, dynamic item set counting algorithm, algorithm for mining generalized and multi-level association rules, and algorithm for mining sequential patterns [Chen et al. 96] [Ganti et al. 99].

In this module, the Apriori algorithm [Agrawal and Srikant 94] is used for mining association rules. In particular, it is used to discover all association rules, which show some interesting characteristics of the software components mapped onto a particular area of the GHSOM or the same growing SOM.

The main idea behind the Apriori algorithm is to scan the transactional database to search for *k*-itemsets (*k* items belonging to the set of items *I*). As the name of the algorithm implies, it uses prior knowledge for discovering large item sets in the database. The algorithm performs iteratively and uses the *k*-itemsets discovered to find the *(k+1)*-itemsets. In each iteration, the algorithm performs in three steps. First, it produces a candidate set of large itemsets. Then, it counts the number of occurrences of each candidate itemset. Finally, it determines large itemsets based on a predetermined minimum support. If no new large itemsets are found, it terminates. The Apriori algorithm and its supporting function, named the apriori-gen function, are shown in Figures 3.5 and 3.6, respectively.

```
1)  L1 = {large 1-itemsets};
2)  for ( k = 2; Lk-1 ≠ ∅; k++) do begin
3)     Ck = apriori-gen(Lk-1);     // new candidates
4)     for all transactions t ∈ D do begin
5)        Ct = subset(Ck, t);     // candidates contained in t
6)        for all candidates c ∈ Ct do
7)           c.count++;
8)     end
9)     Lk = { c ∈ Ck | c.count ≥ minsup}
10) end
11) Answer = ∪k Lk;
```

Figure 3.5 The Apriori algorithm [Agrawal and Srikant 94]

```
// join L_{k-1} with L_{k-1}
insert into C_k
select  p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}
from  L_{k-1} p, L_{k-1} q
where  p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2},
       p.item_{k-1} < q.item_{k-1};

// delete all itemset c ∈ C_k such that some (k-1)-subset of c
// is not in L_{k-1}
for all itemsets c ∈ C_k do
    for all (k-1)-subsets s of c do
        if (s ∉ L_{k-1}) then
            delete c from C_k;
```

Figure 3.6 The apriori-gen function [Agrawal and Srikant 94]

Let consider an example of how the Apriori algorithm works in each pass, as illustrated in Figure 3.7. In the first pass, the set of candidate *1*-itemsets, C_1, is easily obtained. Assume that the minimum support is 40% or two transactions. The set of large *1*-itemsets, L_1, containing candidate *1*-itemsets with the minimum support required, can be determined. In the second pass, the set of candidate *2*-itemsets, C_2, is built from L_1. The set of large *2*-itemsets, L_2, is determined based on the support of each candidate *2*-itemset in C_2.

In the third pass, the set of candidate *3*-itemsets, C_3, is constructed from L_2 as follows. First, two large *2*-items with the same first item, such as $\{BC\}$ and $\{BE\}$, are identified. Then, it tests whether the *2*-itemset $\{CE\}$, which consists of their second items, is a large *2*-itemset or not. Since $\{CE\}$ is also a large itemset, meaning that all the subsets of $\{BCE\}$ are large. Therefore, $\{BCE\}$ becomes a candidate *3*-itemset. After that

the set of large *3*-itemsets, L_3, is discovered. Since there is no candidate *4*-itemset to be constructed from L_3, the Apriori algorithm ends [Chen et al. 96].

By using the large itemsets found, all association rules are generated straightforwardly as follow. For every large itemsets L, find all subsets of L, say X. Check if a rule of the form $X \Rightarrow (L\text{-}X)$ holds by calculating the ratio *conf* = support(L)/support(X). If *conf* $\geq$ *minconf*, then the rule holds [Agrawal and Srikant 94]. Considering our example, *BCE* and *BC* are large itemsets and *BC* is a subset of *BCE*, suppose that the ratio *conf* = support(*BCE*)/support(*BC*) $\geq$ *minconf*. The rule $BC \Rightarrow E$ is generated.

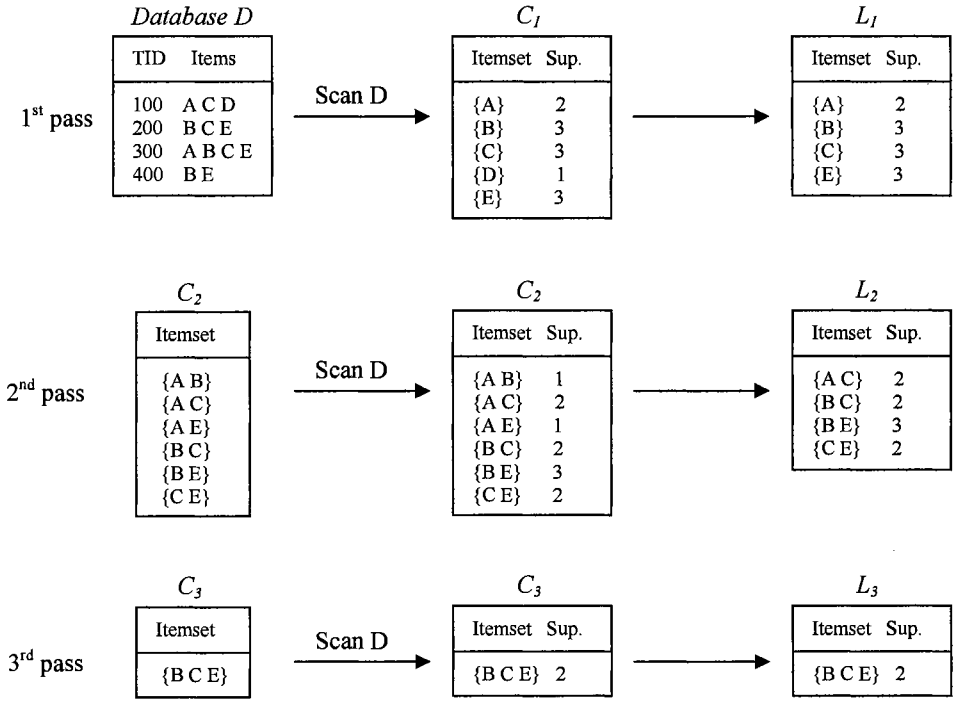


Figure 3.7 How the Apriori algorithm works in each pass [Chen et al. 96]

3.5 Visualization and Retrieval

Two tools are provided for software developers to search for desired software components to be used in a reuse-driven development environment.

- *Browsing Tool:* Developers can explore the resulting GHSOM, look around among similar software components, and discover unanticipated opportunities for reuse. Any subarea of the maps can be selected in order to move from one layer to another layer. For the last layer, by clicking the neuron, a set of reusable candidates along with their corresponding association rules are presented.
- *Keyword Searching Tool:* Developers can confine their attention to a certain group of software components by specifying a set of keywords expressing their requirements. Instead of starting from the first layer, the developers can start from any map at any layer. This tool is useful when developers know what kind of software components they are looking for.

CHAPTER IV

EXPERIMENTS AND RESULTS

4.1 Experiment Objectives

The experiments were conducted with two primary objectives:

1) To demonstrate the potential of the GHSOM for the organization and visualization of a collection of reusable components stored in a software repository, and compare the results with the ones obtained by using the traditional SOM.
2) To show the usefulness of the mining association rules for the discovery of some interesting characteristics about reusable components, which are mapped onto the same particular area of the GHSOM.

4.2 Data Sets

There are five data sets used in this study. Each data set consists of several hundreds of C/C++ program source code files, which are collected from many websites. Table 4.1 provides information about the data sets including the brief description, the number of files, the number of lines of comment, the number of lines of code, and the ratio of comment lines per code lines. The lengths of the sample programs in the

collection range from a few to several hundred lines of code. For the detail lists of the files in the data sets can be found in Appendix B.

Table 4.1 The data sets

	Description	No of Files	Lines of Comment	Lines of Code	Ratio Comment/Code
Data Set 1	DS, IR, AI	273	13481	15394	0.88
Data Set 2	Machine Learning C++	351	18722	32840	0.57
Data Set 3	GNU Scientific Library	998	23959	126318	0.19
Data Set 4	AR, SV, GA, FL, NN, DT	413	25093	60665	0.41
Data Set 5	Data Set 2 and Data Set 4	764	43815	93505	0.47

- Data Set 1

Data Set 1 was gathered from three well-known textbooks that are widely used as major references in computer science classes. The titles, author names, and URLs for downloading the program source code files of the textbooks are listed below.

1) Data Structures and Algorithm Analysis in C++ by Mark Allen Weiss

(http://www.cs.fiu.edu/~weiss/dsaa_c++/code/)

2) Information Retrieval Data Structures & Algorithms by Bill Frakes

(http://www.dcc.uchile.cl/~rbaeza/iradsbook/irbook.html)

3) Artificial Intelligence: A Modern Approach by Stuart Russell and Peter Norvig

(http://www.cs.berkeley.edu/~russell/aima.html)

Data Set 1 consists of 273 files in total. 110 files are from the first source, 47 files are from the second source, and 116 files are from the third source.

- Data Set 2

Data Set 2, the Machine Learning C++ (MLC++), is a public domain software library of C++ classes for supervised machine learning, originally developed at Stanford University and now being distributed by Silicon Graphics, Inc. (SGI) [Kohavi et al. 96]. The library provides general machine learning algorithms with a wide variety of tools that can help mine data, accelerate development of new mining algorithms, provide comparison tools, and display information visually. 351 files were chosen as Data Set 2. The URL for downloading the MLC++ is at http://www.sgi.com/tech/mlc/source.html.

- Data Set 3

Data Set 3 is the GNU Scientific Library (GSL), a numerical library for C and C++ programmers. It is free software under the GNU General Public License. The library provides a wide range of mathematical routines such as vector and matrix manipulation, random number generators, special functions, statistics, and least-squares fitting. The total of 998 files was selected to be population of Data Set 3. The GSL can be obtained from the URL at http://www.gnu.org/software/gsl.

- Data Set 4

Data Set 4 consists of six categories of programs related to six notable data mining algorithms and techniques including association rules, support vector machine, genetic algorithms, fuzzy logic, neural network, and decision tree, which were gathered from the following URLs.

1) Association Rules (AR)

- http://www.cs.bme.hu/~bodon/en/apriori/apriori.tar.gz
- http://fuzzy.cs.uni-magdeburg.de/~borgelt/apriori.html
- http://fuzzy.cs.uni-magdeburg.de/~borgelt/eclat.html
- http://db.cs.helsinki.fi/~goethals/cgi-bin/apriori.tgz
- http://db.cs.helsinki.fi/~goethals/cgi-bin/dic.tgz
- http://db.cs.helsinki.fi/~goethals/cgi-bin/eclat.tgz
- http://db.cs.helsinki.fi/~goethals/cgi-bin/fpgrowth.tgz
- http://db.cs.helsinki.fi/~goethals/cgi-bin/rules.tgz

2) Support Vector Machine (SV)
   - http://five-percent-nation.mit.edu/SvmFu/
   - http://www.csie.ntu.edu.tw/~cjlin/libsvm/
   - http://www.cs.cornell.edu/People/tj/svm_light/
   - http://www.idiap.ch/learning/SVMTorch.html
3) Genetic Algorithms (GA)
   - http://lancet.mit.edu/galib-2.4/
   - ftp://ftp-illigal.ge.uiuc.edu/pub/src/ECGA/Cpp/ECGA.tar.Z
   - ftp://ftp-illigal.ge.uiuc.edu/pub/src/LLGA/Cpp/LLGA.tar.Z
4) Fuzzy Logic (FL)
   - http://ffll.sourceforge.net/downloads.htm
5) Neural Network (NN)
   - http://prokop.ae.krakow.pl/projects/nnlib.html
   - http://sourceforge.net/projects/nnetlib/
   - http://sourceforge.net/projects/inanna/

6) Decision Tree (DT)

- http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz
- http://k2.nimkathana.com/~gkt/export/CPDC/cn2.tar.gz
- http://mow.ecn.purdue.edu/~brodley/software/lmdt.html

Data Set 4 is made up of 413 files, including 68 AR files, 57 SV files, 115 GA files, 42 FL files, 49 NN files, and 82 DT files.

- Data Set 5

Data Set 5 is a combination of Data Set 2 and Data Set 4, which have similar characteristics in terms of the application domain and the ratio of comment/code.

4.3 Software Tools and Computer Systems Used

4.3.1 Software Tools Used

A number of software tools were used to perform the experiments. The functions of the tools and their URLs are given below.

- Stopwords lists for eliminating "stopwords" from a program source code file.

  (http://www.onjava.com/onjava/2003/01/15/examples/EnglishStopWords.txt)
- Porter stemmer program in Java for grouping words that have the same common linguistic roots in a program source code file.

  (http://www.tartarus.org/~martin/PorterStemmer/java.txt)
- SOMLib Java package for extracting keywords from a collection of program source code files and creating the feature vectors, which are served as input vectors for the construction of the SOM and the GHSOM.

(http://www.ifs.tuwien.ac.at/~andi/somlib/)

- SOM Toolbox for MATLAB for creating the SOM [Vesanto et al. 00].

  (http://www.cis.hut.fi/projects/somtoolbox)

- GHSOM Toolbox for MATLAB for creating the GHSOM [Chan and Pampalk 02].

  (http://www.ifs.tuwien.ac.at/~andi/ghsom)

- MATLAB Version 6.1.0.450 Release 12.1 for running SOM Toolbox and GHSOM Toolbox

  (http://www.mathworks.com/)

- Understand for C++ tool for extracting software metrics, e.g., lines of code, lines of comments, and source code items, i.e., lists of all include files. These set of items were used for mining association rules.

  (http://www.scitools.com/)

- Apriori program for mining association rules [Borgelt 04].

  (http://fuzzy.cs.uni-magdeburg.de/~borgelt/apriori.html)

4.3.2 Computer Systems Used

Two computer systems were used to run the programs in the experiments.

1) The CSA machine operated by the Computer Science Department at Oklahoma State University. It is the Sunfire v880 with two UltraSparc 3+ 900 MHz processors, 4GB of RAM, about 120 GB of disk space, and Solaris 9. The CSA was used to run the Stopwords elimination program, Porter stemmer program, SOMLib Java package, and the Apriori program.

2) A personal computer. Its specification is Intel Pentium 4 CPU 2.80 GHz, 512 MB of RAM, about 80 GB of disk space, and Microsoft Windows XP Professional Version 2002 Service Pack 2. The personal computer was used to run the MATLAB application, SOM Toolbox, GHSOM Toolbox, and Understand for C++ tool.

4.4 Results

4.4.1 Feature Vectors for the Construction of SOM and GHSOM

To investigate the effect of data preprocessing (including eliminating "stopwords" and stemming) to the number of features extracted, two experiments – feature extraction without preprocessing and feature extraction with preprocessing – were conducted on each data set. Assume that the minimum word length is 3, minimum component frequency for a word to be selected as a feature is 0.02 or 2%, and maximum component frequency for a word not to be removed is 0.8 or 80%.

Table 4.2 shows the results of the experiments by comparing the number of words extracted, removed, and selected as features. The results show that the number of features extracted with preprocessing is less than the one extracted without preprocessing by the average of 27.4 %. Hence, it is suitable to use the features extracted with preprocessing to create feature vectors for representing the reusable components because the preprocessing can help obtain high-quality features when dealing with a large number of software components.

Based on the features extracted with preprocessing, feature vector files, which served as input vector spaces for the construction of SOM and GHSOM, were prepared

according to the input file format of the SOM Toolbox and the GHSOM Toolbox. Some sample pages of a feature vector file for Data Set 1 can be found in Appendix C.

Table 4.2 Feature extraction without and with preprocessing

	# files	% less	Without preprocessing			With preprocessing		
			words	removed	features	words	removed	features
Data Set 1	273	29.5	2801	2032	769	1869	1327	542
Data Set 2	351	31.2	6460	5270	1190	4652	3833	819
Data Set 3	998	18.6	4589	4106	483	3492	3099	393
Data Set 4	413	28.4	8233	6845	1388	6158	5164	994
Data Set 5	764	29.3	12149	10815	1334	9287	8344	943

4.4.2 Comparison of the SOM and the GHSOM

The results of the SOM and the GHSOM were analyzed in three different perspectives: 1) visualization of the resulting maps, 2) structure of the resulting maps, and 3) training time, which are explained in the following three subsections.

4.4.2.1 Visualization of the Resulting Maps

In order to compare the visualization of the resulting maps of the SOM and the GHSOM, two primary experiments were conducted for each data set. The first experiment is to construct a number of SOMs by varying the map size. The second experiment is to construct a number of GHSOMs by varying the values of the thresholds τ_1 and τ_2. Some sample figures of the resulting maps for Data Set 1, 2, 3, 4, and 5 were exhibited respectively.

- Data Set 1

A 10×10 SOM for Data Set 1 was produced, as depicted in Figure 4.1, and its U-matrix (unified distance matrix) with interpolated shading of colors is displayed in Figure 4.2. By inspecting Figure 4.1 visually, it is noticeable that there are three main groups of software components which are Data Structure (DS), Information Retrieval (IR), and Artificial Intelligence (AI), as labeled with cluster titles. Software components with similar features are apparently located on nearby regions of the map. For example in IR, a cluster of Stemmer programs can be found at the right side of the map, and next to it is a cluster of Stopper programs. As another example, a cluster of Thesauri programs shows up at the bottom right corner of the map.

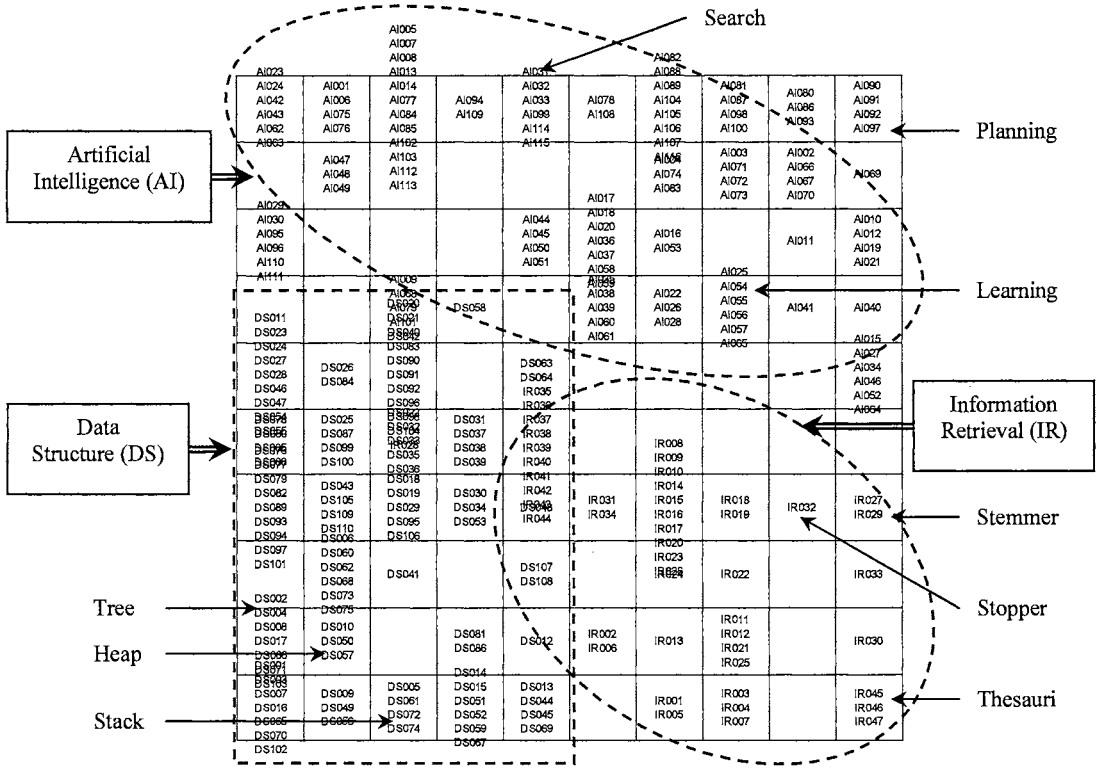


Figure 4.1 The resulting 10×10 SOM for Data Set 1

The U-matrix presented in Figure 4.2 offers a better way to get insight of the data distribution. It is helpful to visualize distance between neighboring map units, which is calculated and presented with different colors, and hence reveals the cluster structure of the map. High values corresponding to a large distance indicate a cluster border, while uniform areas of low values indicate clusters themselves [Vesanto et al. 00]. As seen in the figure, a DS cluster is at the left side of the map, an AI cluster is at the upper right of the map, and an IR cluster is at the lower right of the map.

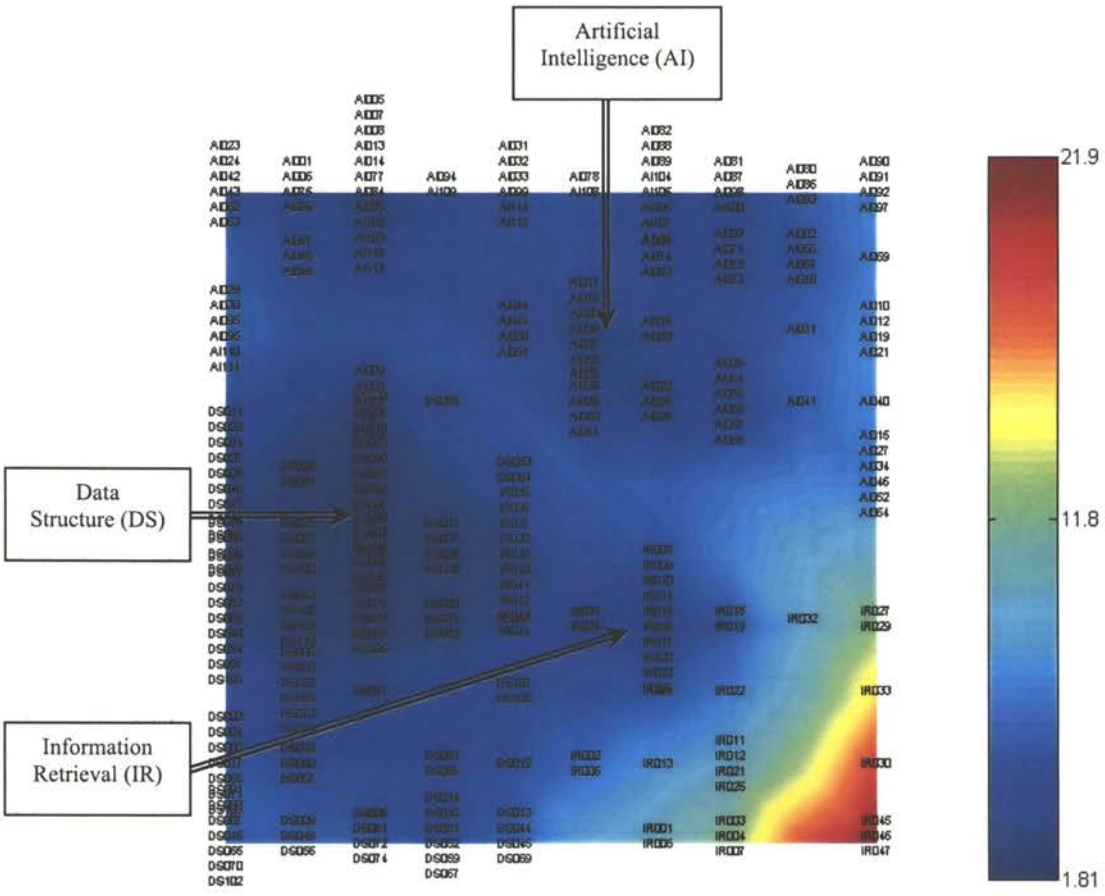


Figure 4.2 The U-matrix of the 10×10 SOM for Data Set 1

A 4-layer GHSOM for Data Set 1 was generated by setting the thresholds τ_1 = 0.8500 and τ_2 = 0.0035, as illustrated in Figure 4.3. It can be seen that the clusters are the areas with high data densities on the map that are further hierarchically expanded by growing SOMs. In the figure, the top layer maps are depicted in gray and the bottom layer maps are depicted in white. The first layer map, consisting of 3×3 neurons, shows the three major clusters of software components: DS, IR, and AI. Most neurons of the first layer SOM have been expanded in the second layer maps. Two of its submaps are presented in Figure 4.4.

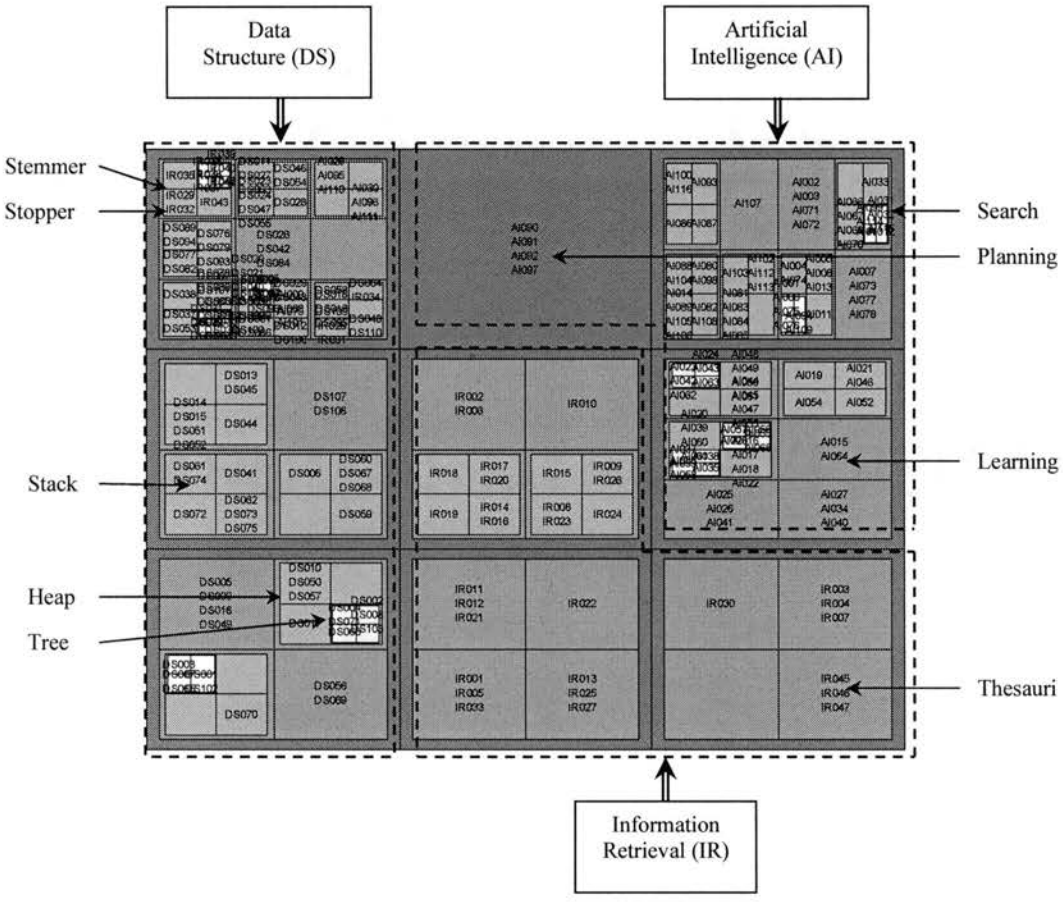
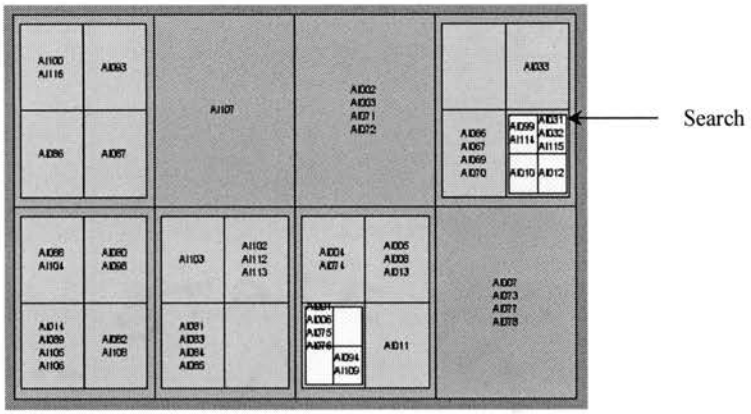
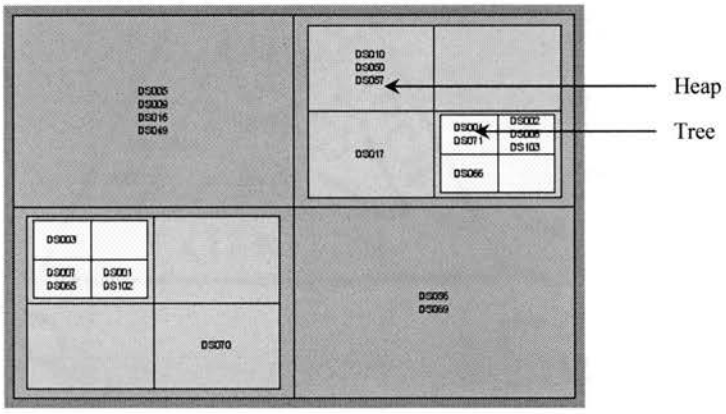


Figure 4.3 The resulting 4-layer GHSOM for Data Set 1



(a) The AI submap



(b) The DS submap

Figure 4.4 Two submaps of the resulting 4-layer GHSOM for Data Set 1

Figure 4.4(a) shows an AI submap, consisting of 2×4 neurons, expanded from the neuron at the upper right corner (row 1 and column 3). Figure 4.4(b) shows a DS submap, consisting of 2×2 neurons, expanded from the neuron at the lower left corner (row 3 and column 1). Let's have a look at Figure 4.4(b). Programs related to Tree structure, e.g., AVL Tree, Binary Search Tree, and Splay Tree are grouped together at the upper right of the map, and programs related to Heap structure, e.g., Leftist Heap, Pairing Heap, and Treap are grouped separately in the same vicinity.

- Data Set 2

Figure 4.5 displays a 15×15 SOM for Data Set 2 and Figure 4.6 shows its U-matrix representation. As seen in Figure 4.5, several major categories of MLC++ components such as ML (ML), MFSS (MF), Include (IN), MTrans (MT), and MCore (MC) are designated with cluster titles. A group of Include files can be found at the upper right and a group of Mcore files can be found at the lower right. In the middle of the map is where a group of ML and MFSS files located.

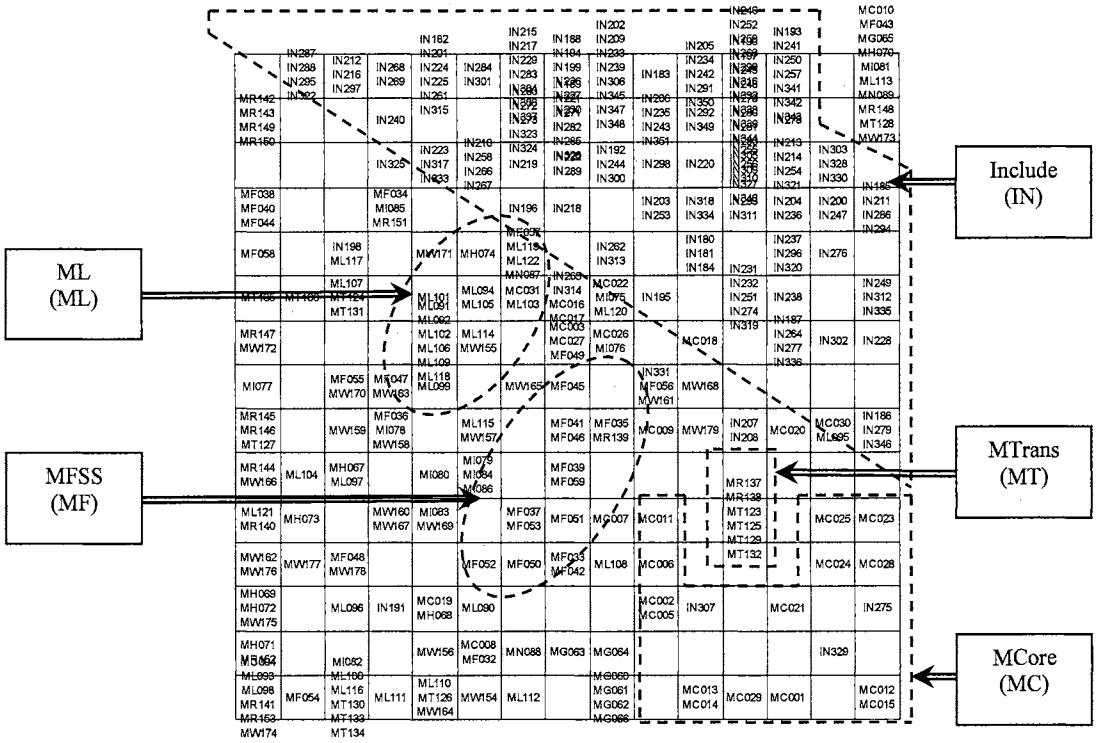


Figure 4.5 The resulting 15×15 SOM for Data Set 2

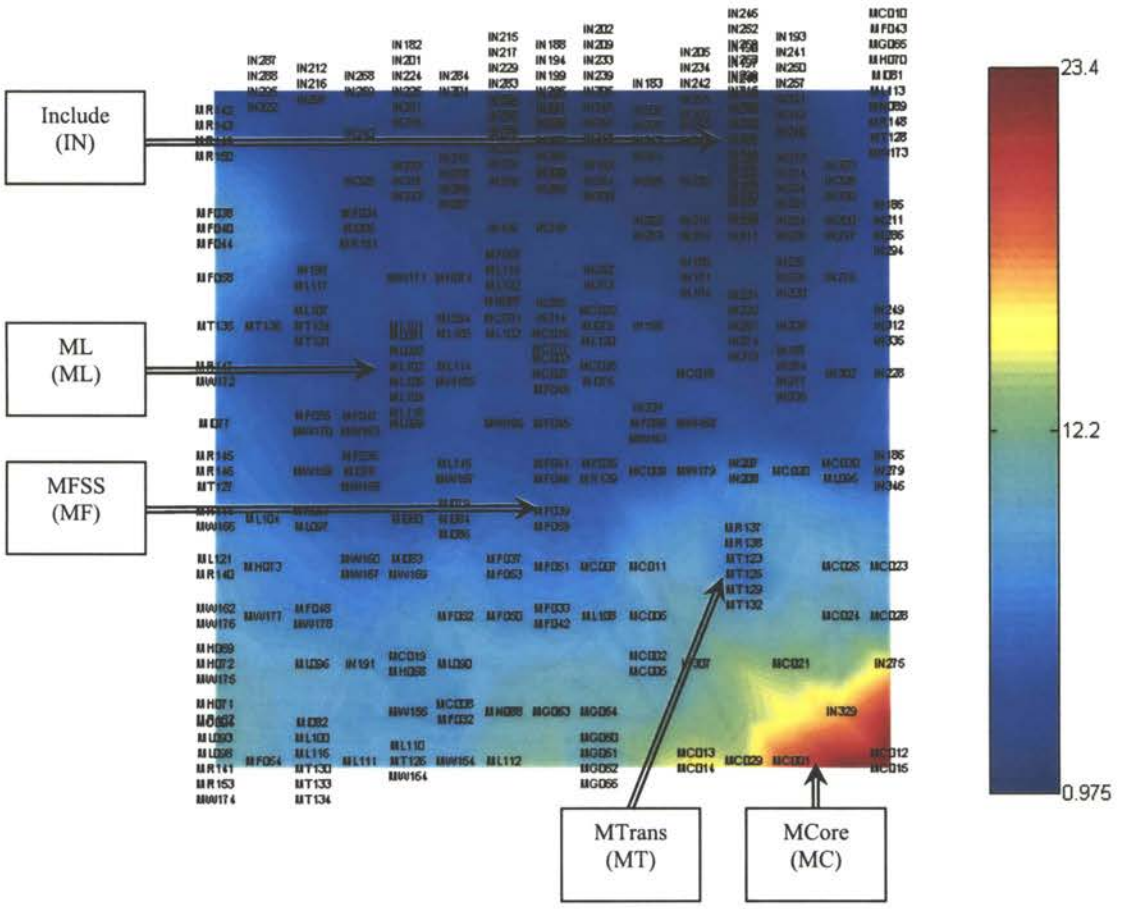


Figure 4.6 The U-matrix of the 15×15 SOM for Data Set 2

A 5-layer GHSOM for Data Set 2 was generated by setting the thresholds τ_1 = 0.8500 and τ_2 = 0.0035, as illustrated in Figure 4.7. The first layer map, consisting of 7×5 neurons, shows several major clusters of MLC++ components: ML, MF, IN, MT, and MC. Many neurons of the first layer SOM have been expanded in the second layer maps. Two of its submaps are presented in Figure 4.8. Figure 4.8(a) shows a MF submap, consisting of 2×2 neurons, expanded from the neuron at the upper left corner (row 1 and column 1). Figure 4.8(b) shows an IN submap, consisting of 2×3 neurons, expanded from the neuron at the upper right corner (row 1 and column 5).

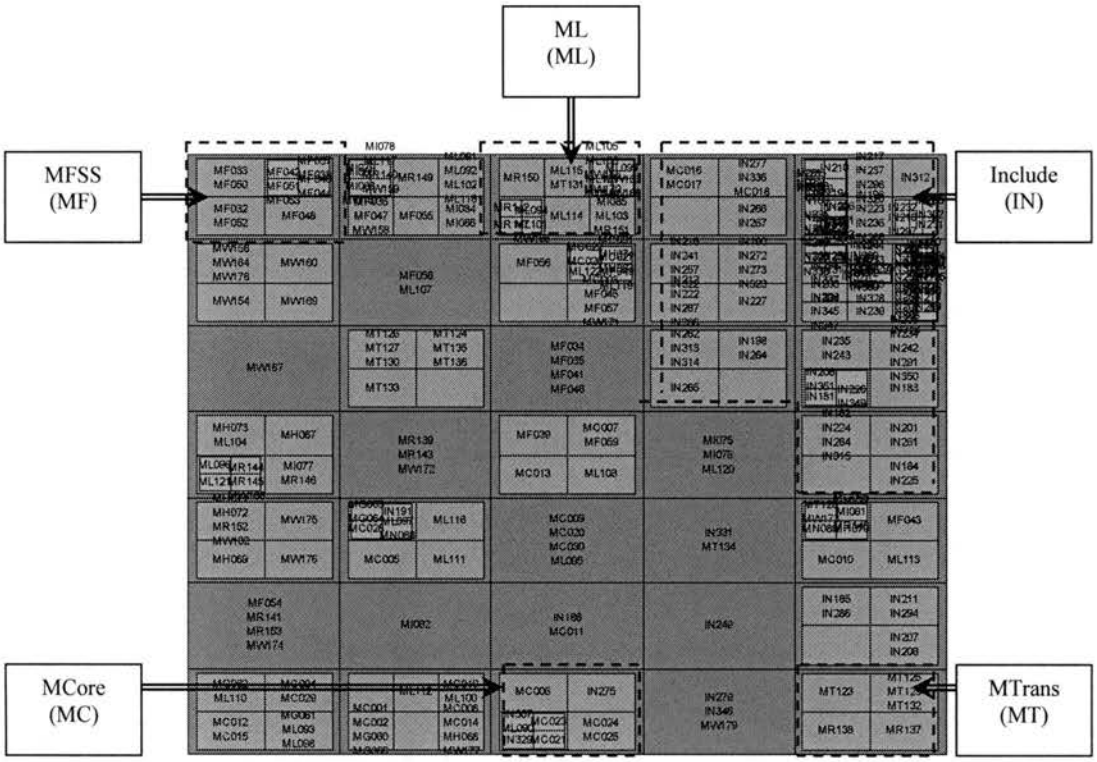
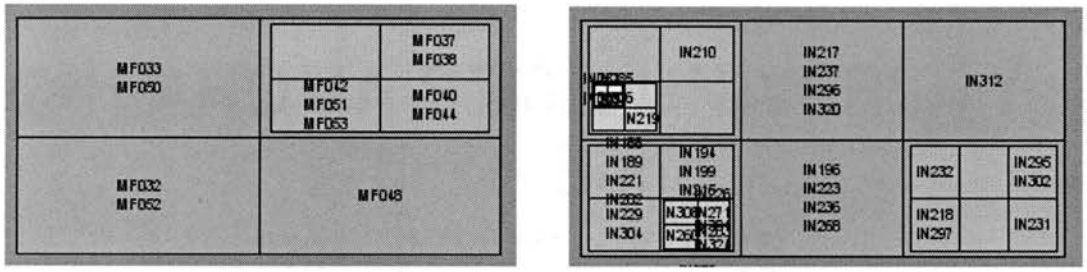


Figure 4.7 The resulting 5-layer GHSOM for Data Set 2



(a) The MF submap (b) The IN submap

Figure 4.8 Two submaps of the resulting 5-layer GHSOM for Data Set 2

- Data Set 3

Figure 4.9 displays a 30×30 SOM for Data Set 3 and Figure 4.10 shows its U-matrix representation. As seen in Figure 4.9, several major categories of GSL components such as Sorting (SO), Permutation (PE), Statistics (ST), Random Number Generation (RN), Ordinary Differential Equation (OD), CBLAS (CB), Fast Fourier Transforms (FF), and Special Functions (SP) are identified with cluster titles. A group of CBLAS files can be found at the upper right and below it are groups of FF and SP files.

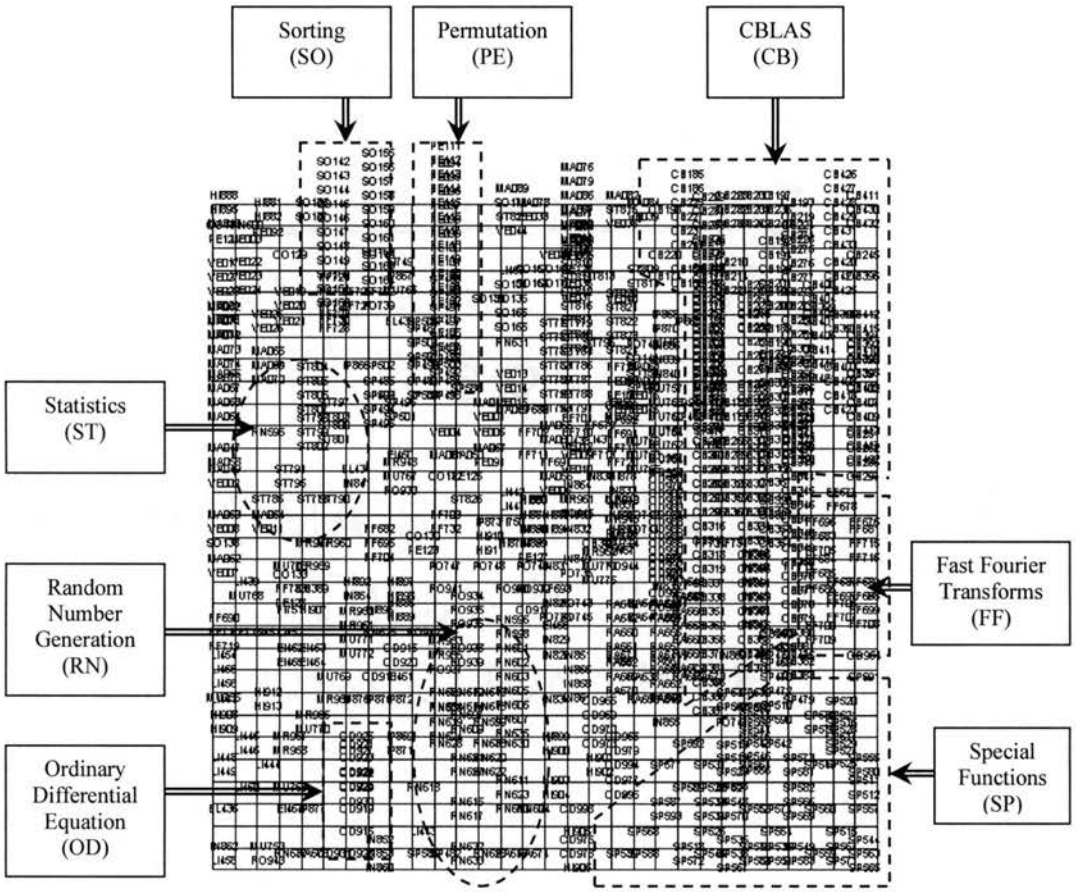


Figure 4.9 The resulting 30×30 SOM for Data Set 3

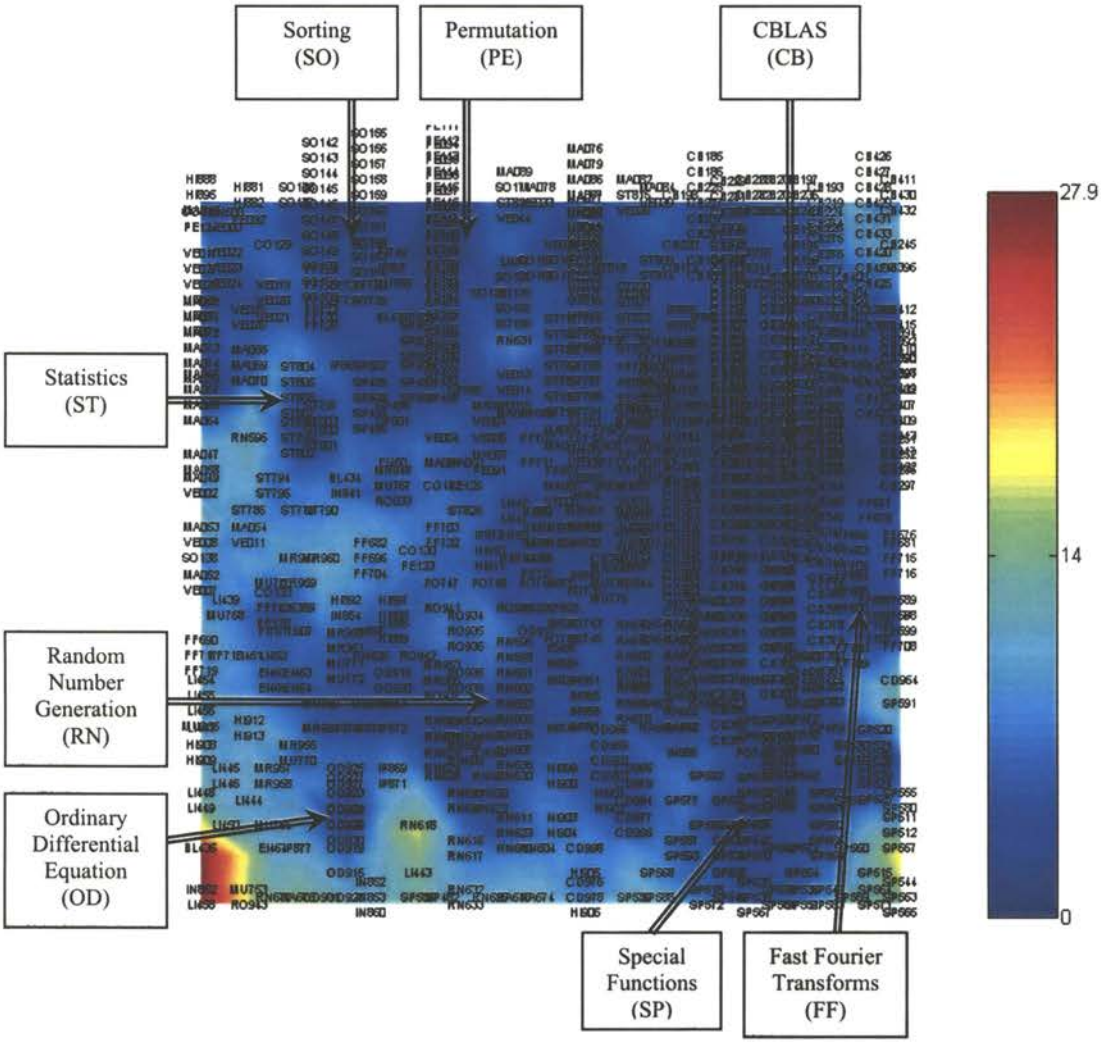


Figure 4.10 The U-matrix of the 30×30 SOM for Data Set 3

A 5-layer GHSOM for Data Set 3 was generated by setting the thresholds τ_1 = 0.8500 and τ_2 = 0.0035, as illustrated in Figure 4.11. The first layer map, consisting of 6×3 neurons, shows several major clusters of GSL components: SO, PE, ST, RN, OD, CB, FF, and SP. Most neurons of the first layer SOM have been expanded in the second layer maps. Two of its submaps are presented in Figure 4.12. Figure 4.12(a) shows a ST submap, consisting of 2×2 neurons, expanded from the neuron at row 3 and column 2.

Figure 4.12(b) shows a FF submap, consisting of 2×2 neurons, expanded from the neuron at row 4 and column 1.

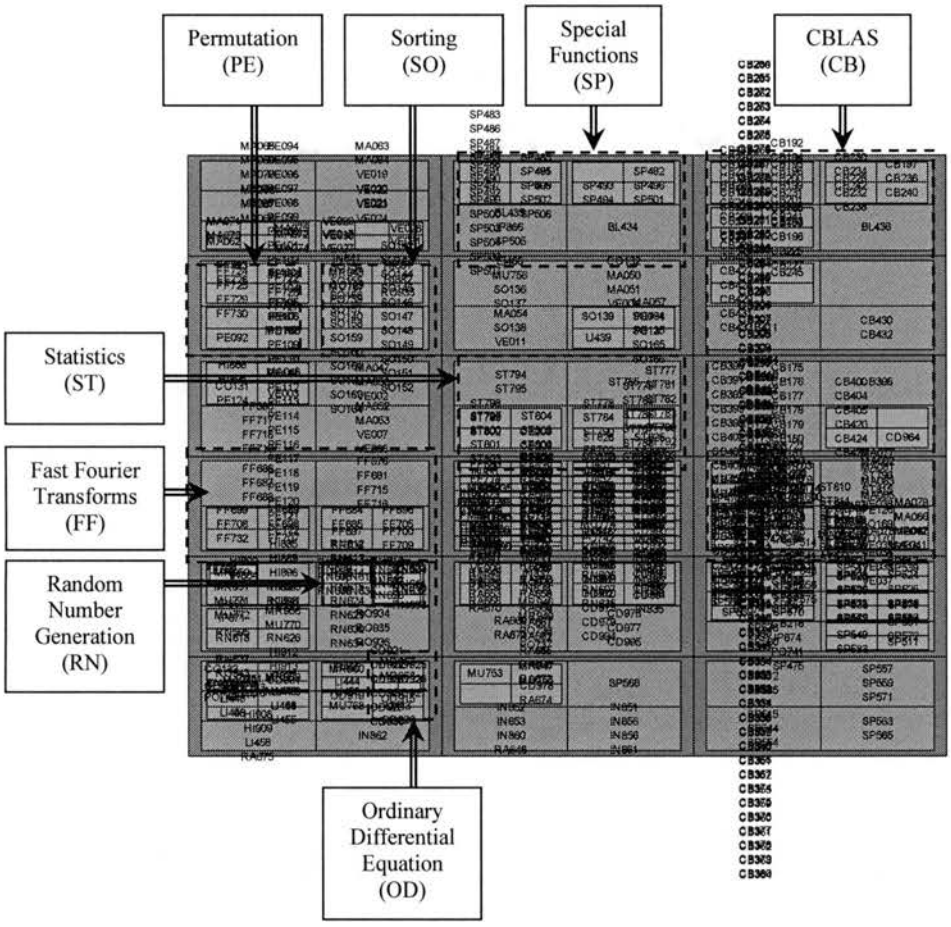
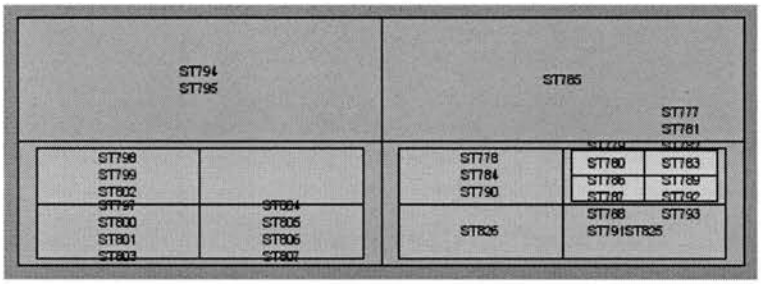
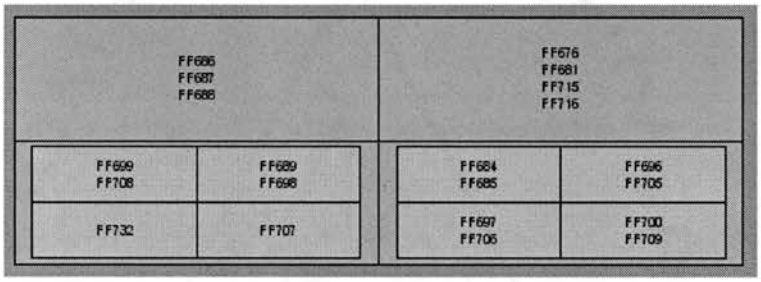


Figure 4.11 The resulting 5-layer GHSOM for Data Set 3



(a) The ST submap



(b) The FF sub map

Figure 4.12 Two submaps of the resulting 5-layer GHSOM for Data Set 3

- Data Set 4

Figure 4.13 displays a 15×15 SOM for Data Set 4 and Figure 4.14 shows its U-matrix representation. As seen in Figure 4.13, there are six major categories of data mining programs, which are Association Rules (AR), Support Vector Machine (SV), Genetic Algorithms (GA), Fuzzy Logic (FL), Neural Network (NN), and Decision Tree (DT), as labeled with cluster titles. The programs related to association rules algorithms were grouped together in an AR cluster at the upper left corner of the map, next to it on the right side is a GA cluster, which contains programs involving genetic algorithms, and below it is a FL cluster, consisting of programs concerning fuzzy logic techniques.

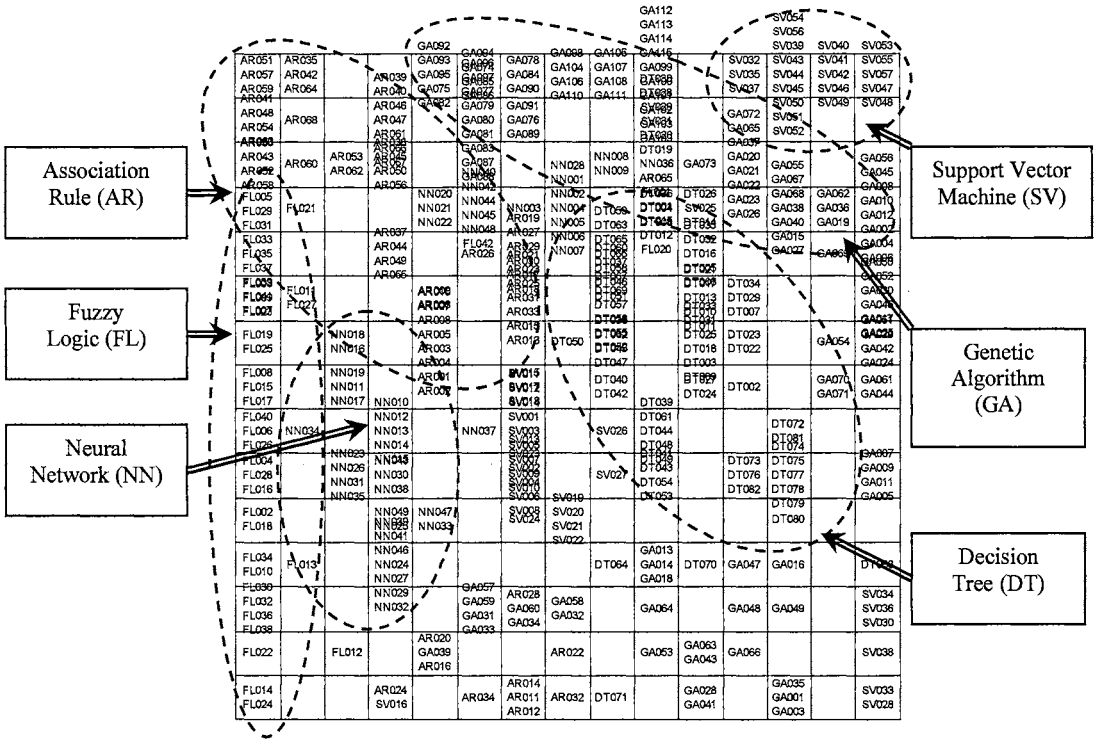


Figure 4.13 The resulting 15×15 SOM for Data Set 4

A 4-layer GHSOM for Data Set 4 was generated by setting the thresholds τ_1 = 0.9000 and τ_2 = 0.0035, as illustrated in Figure 4.15. The first layer map, consisting of 5×3 neurons, shows the six major clusters of software components: AR, SV, GA, FL, NN, and DT. Most neurons of the first layer SOM have been expanded in the second layer maps. Two of its submaps are presented in Figure 4.16. Figure 4.16(a) shows a NN submap, consisting of 2×2 neurons, expanded from the neuron at row 3 and column 1. Figure 4.16(b) shows a FL submap, consisting of 2×2 neurons, expanded from the neuron at row 4 and column 1.

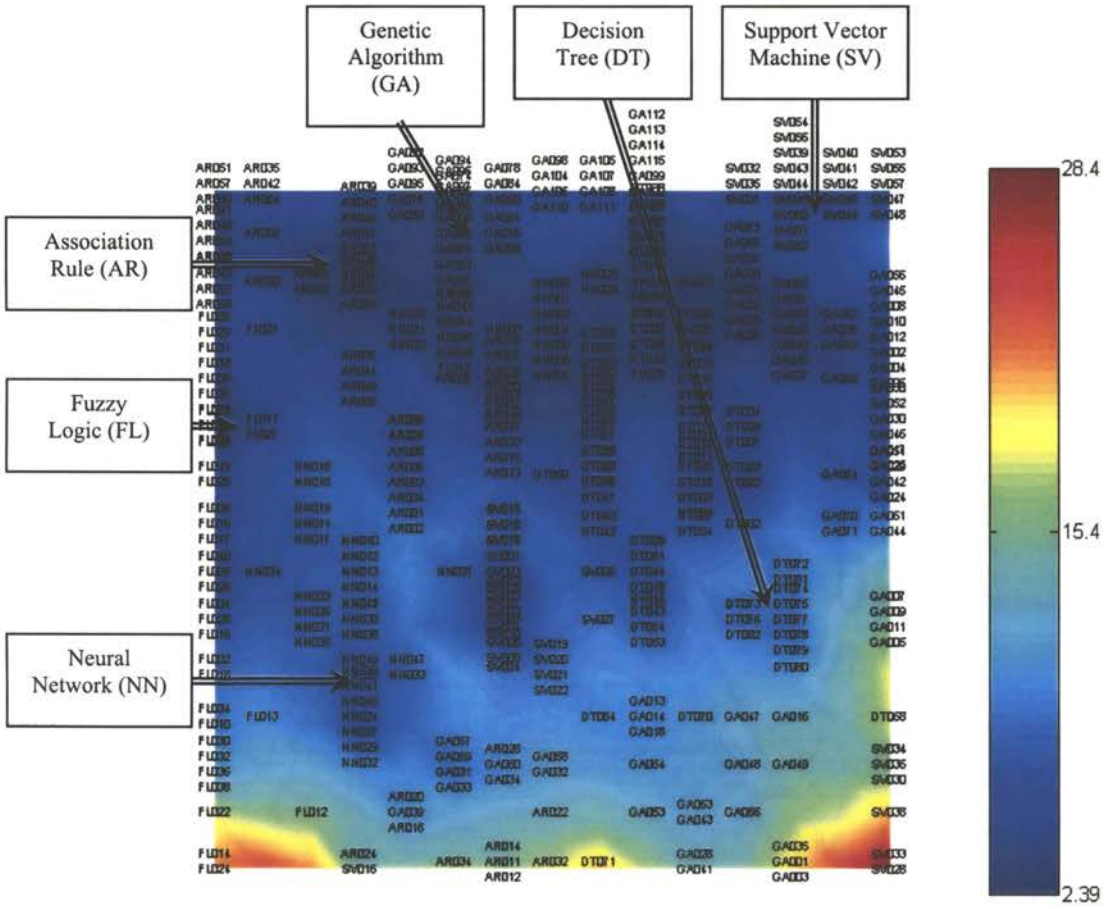


Figure 4.14 The U-matrix of the 15×15 SOM for Data Set 4

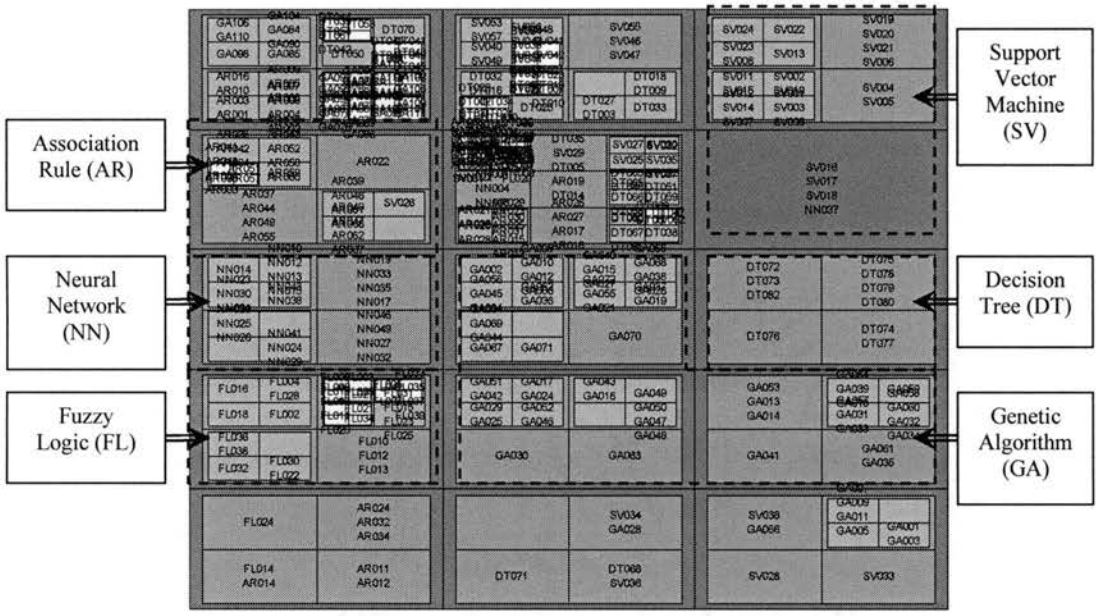
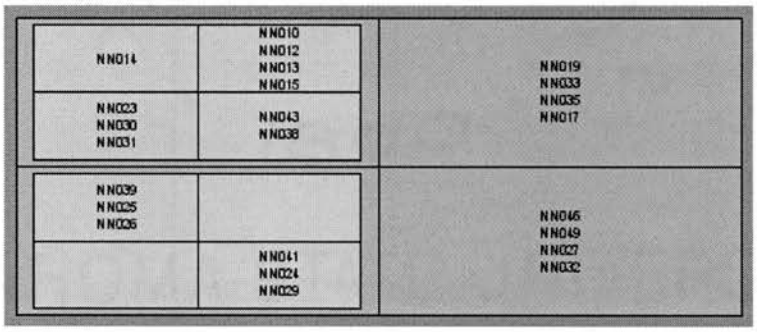
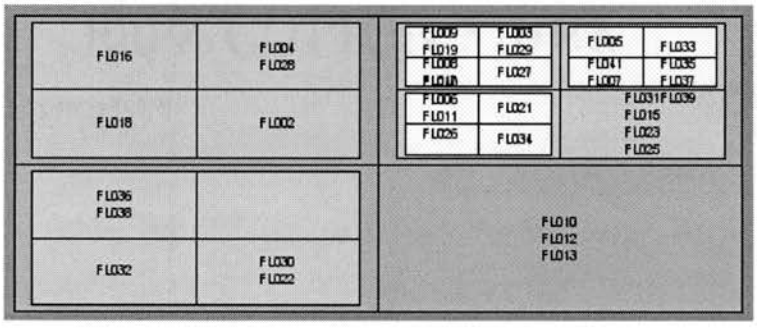


Figure 4.15 The resulting 4-layer GHSOM for Data Set 4



(a) The NN submap



(b) The FL submap

Figure 4.16 Two submaps of the resulting 4-layer GHSOM for Data Set 4

- Data Set 5

Figure 4.17 displays a 20×20 SOM for Data Set 5 and Figure 4.18 shows its U-matrix representation. In Figure 4.17, since Data Set 5 is a combination of Data Set 2 and Data Set 4, software components belonging to Data Set 2 were located in the same neighborhood and separated from software components belonging to Data Set 4.

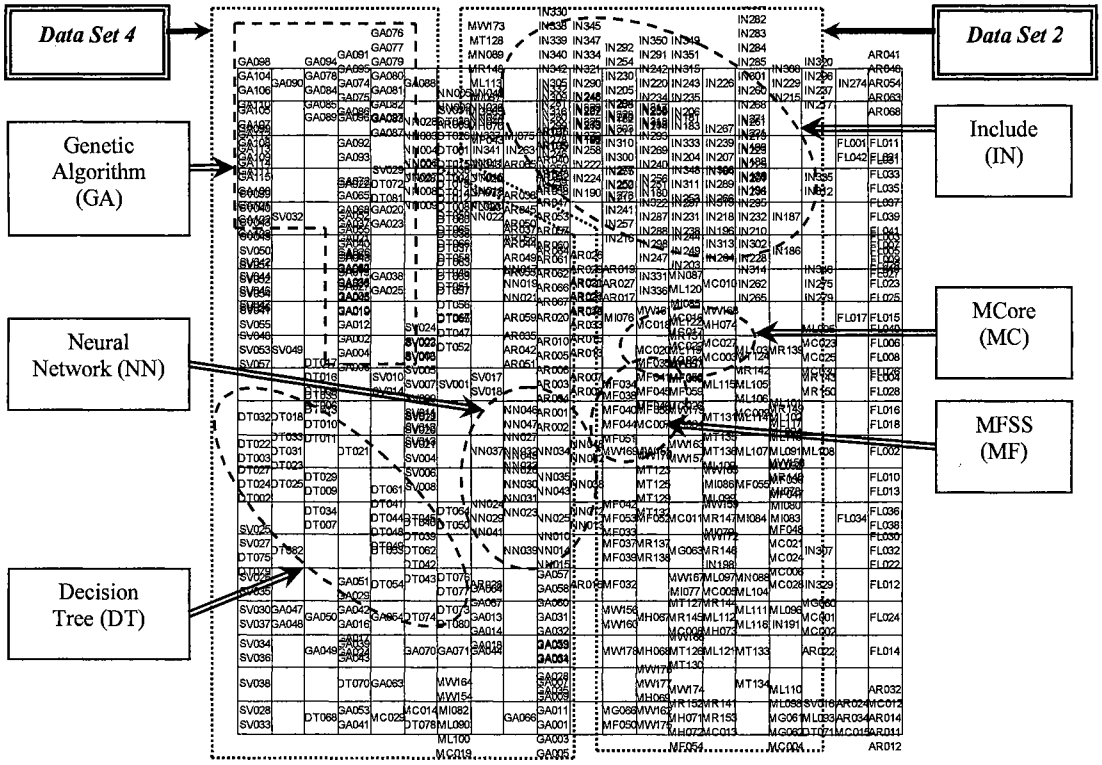


Figure 4.17 The resulting 20×20 SOM for Data Set 5

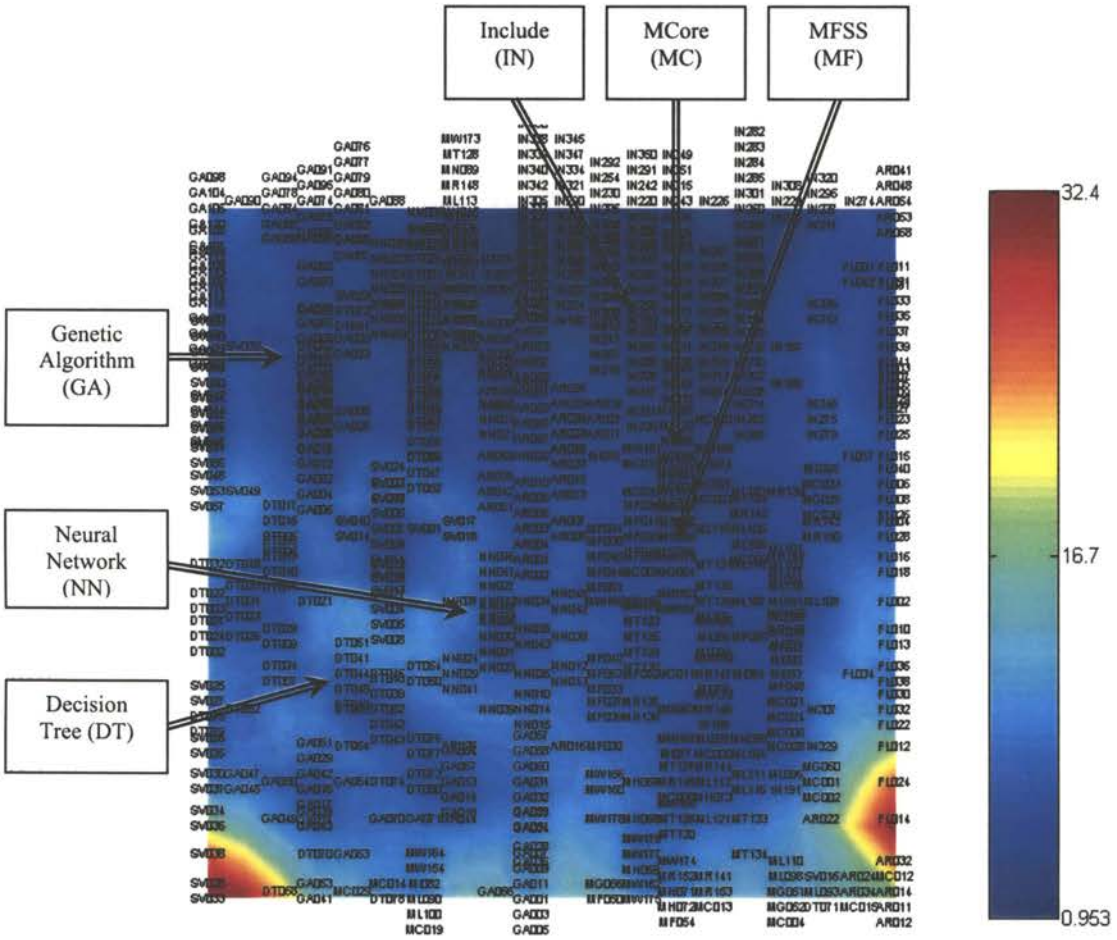


Figure 4.18 The U-matrix of the 20×20 SOM for Data Set 5

A 5-layer GHSOM for Data Set 5 was generated by setting the thresholds τ_1 = 0.9000 and τ_2 = 0.0035, as illustrated in Figure 4.19. The first layer map, consisting of 6×3 neurons, shows the two major clusters of software components: Data Set 2 and Data Set 4. Most neurons of the first layer SOM have been expanded in the second layer maps. Two of its submaps are presented in Figure 4.20. Figure 4.20(a) shows a submap of Data Set 2, consisting of 3×2 neurons, expanded from the neuron at row 3 and column 3.

Figure 4.20(b) shows a submap of Data Set 4, consisting of 2×2 neurons, expanded from the neuron at row 5 and column 1.

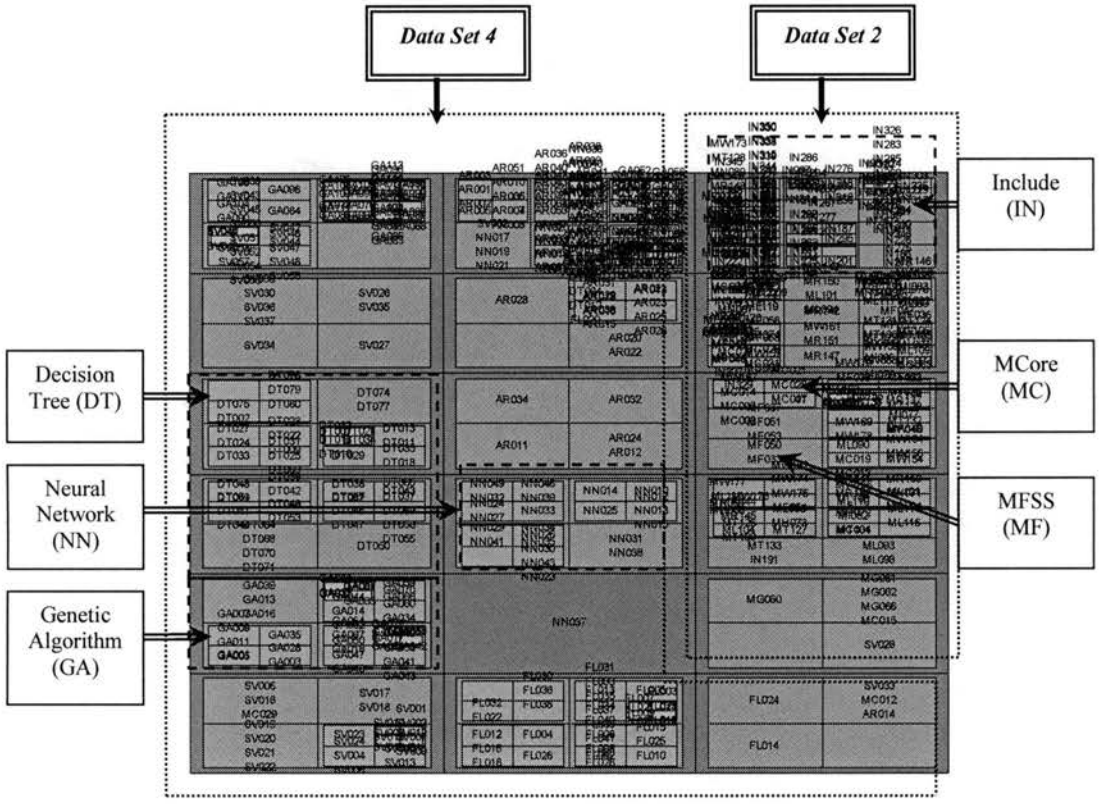
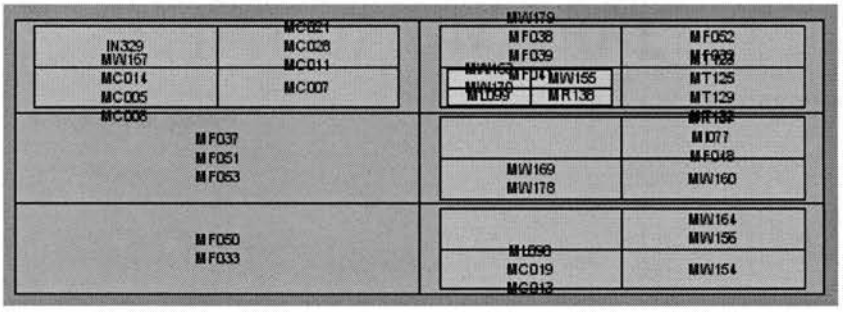
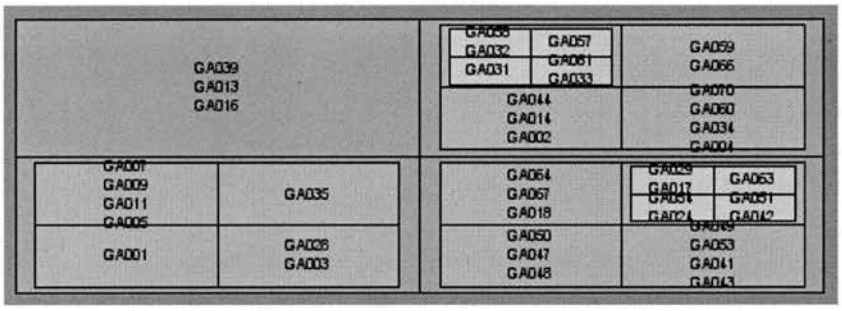


Figure 4.19 The resulting 5-layer GHSOM for Data Set 5



(a) A submap of Data Set 2



(b) A submap of Data Set 4

Figure 4.20 Two submaps of the resulting 5-layer GHSOM for Data Set 5

According to the experimental results, we found that both SOM and GHSOM were successful in creating a topology-preserving representation of the topical clusters of the software components. However, when dealing with a large number of software components, GHSOM behaved better than SOM in the sense that its architecture was determined automatically during its learning process based on the requirement of the input data. Moreover, GHSOM was able to reveal the inherent hierarchical structure of the data into layers and provided the ability to select the granularity of the representation at different levels of the GHSOM.

4.4.2.2 Structure of the Resulting Maps

In this section, structure of the resulting maps of SOM and GHSOM were investigated.

- Structure of SOMs

Typically, the structure of SOMs is evaluated using two quality measures: average quantization error and topology error, as defined below [Vesanto et al. 00].

1) Average quantization error (qe) is the average distance between each input vector and its winning neuron.

2) Topographic error (te) is the percentage of input vectors for which the first and second winning neurons are not adjacent units.

Table 4.3 provides the results of quality (qe and te) of fixed size SOMs with different map size, i.e., 10×10, 15×15, 20×20, 25×25, and 30×30. Table 4.4 provides the results of quality (qe and te) of recommended size SOMs with additional number of rows and columns, i.e., with row+5 and col+5, with row+10 and col+10, and with double row and double column. In this context, the recommended size SOM is a SOM whose map size is determined by the SOM Toolbox using some heuristic formula [Vesanto et al. 00].

Table 4.3 Quality (qe and te) of fixed size SOMs

	SOM 10×10		SOM 15×15		SOM 20×20		SOM 25×25		SOM 30×30	
	qe	te	qe	te	qe	te	qe	te	qe	te
Data Set 1	14.63	0.06	12.25	0.03	10.30	0.03	8.26	0.01	5.80	0.01
Data Set 2	20.30	0.01	18.52	0.03	16.53	0.02	14.44	0.01	11.74	0.02
Data Set 3	11.47	0.11	10.15	0.07	9.16	0.08	8.30	0.06	7.42	0.03
Data Set 4	22.09	0.02	19.78	0.03	17.80	0.02	15.68	0.03	13.15	0.01
Data Set 5	21.53	0.05	20.08	0.03	18.84	0.04	17.57	0.03	16.07	0.02

Table 4.4 Quality (qe and te) of recommended size SOMs

	Recommended Size			With Row+5, Col+5			With Row+10, Col+10			With Row*2, Col*2		
	size	qe	te	size	qe	te	size	qe	te	size	qe	te
Data Set 1	10×8	15.34	0.07	15×13	12.63	0.02	20×18	10.47	0.03	20×16	11.06	0.02
Data Set 2	13×7	20.40	0.02	18×12	18.82	0.01	23×17	16.63	0.01	26×14	17.02	0.01
Data Set 3	14×11	10.74	0.08	19×16	9.61	0.08	24×22	8.58	0.04	28×22	8.31	0.05
Data Set 4	13×8	22.09	0.02	18×13	19.81	0.03	23×18	17.64	0.03	26×16	17.77	0.01
Data Set 5	14×10	21.11	0.06	19×15	19.65	0.03	24×20	18.31	0.03	28×20	17.94	0.03

Graphs presented in Figure 4.21 and 4.22 show the quantization error and topographic error of fixed size SOMs, respectively. Graphs presented in Figure 4.23 and 4.24 show the quantization error and topographic error of recommended size SOMs, respectively. The results show that for every data set, when the map size of SOM increases, qe and te of SOMs trend to decrease.

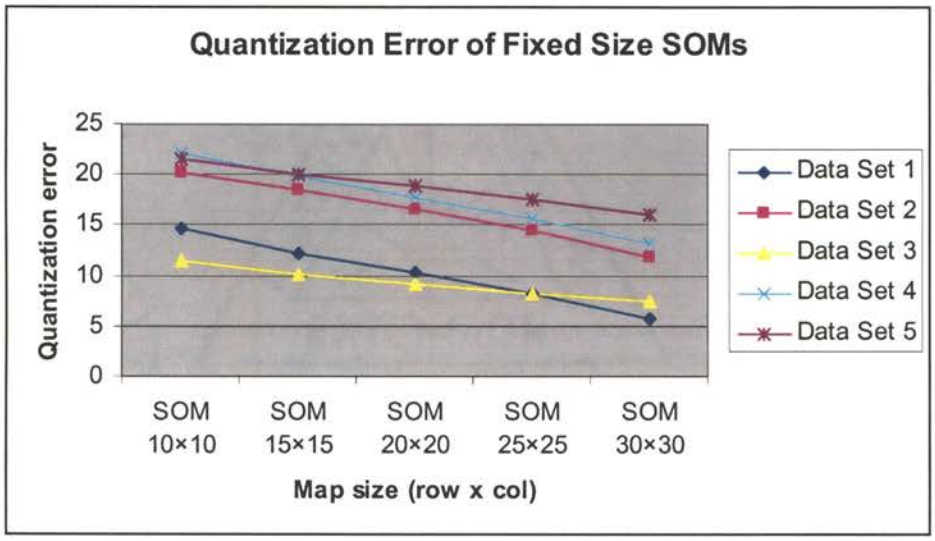


Figure 4.21 Quantization error of fixed size SOMs

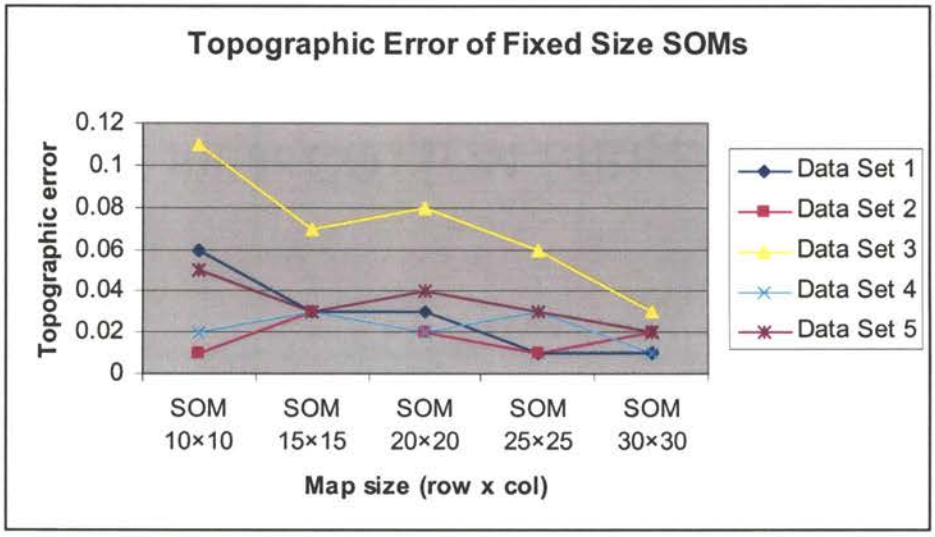


Figure 4.22 Topographic error of fixed size SOMs

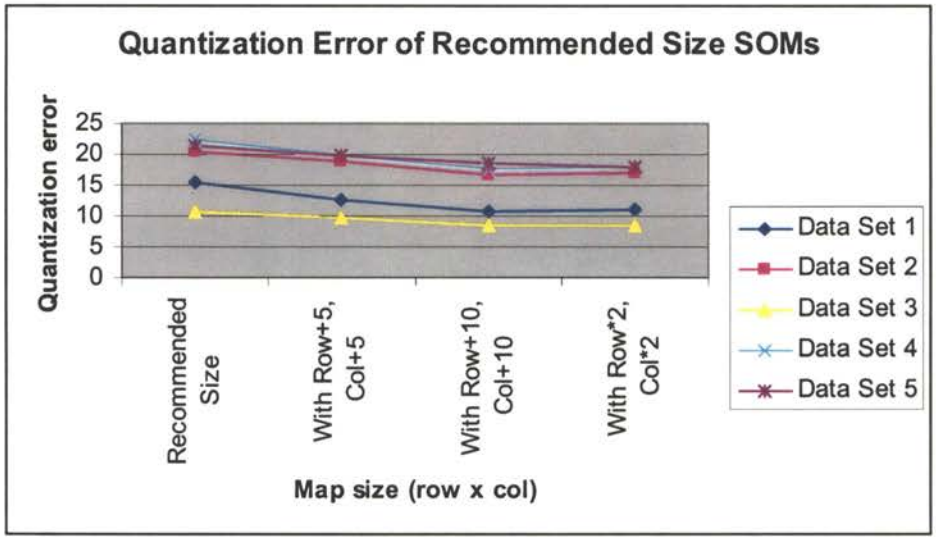


Figure 4.23 Quantization error of recommended size SOMs

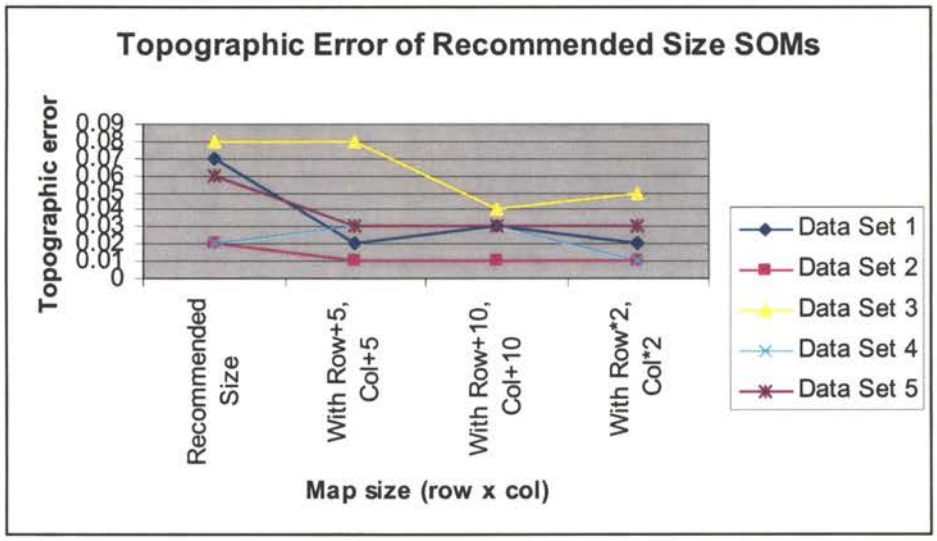


Figure 4.24 Topographic error of recommended size SOMs

- Structure of GHSOMs

The structure of GHSOMs were studied in terms of the number of layers (#L) and the map size at Layer 1 (L1) constructed by varying τ_1 (for controlling breadth of the

maps) and by varying τ_2 (for controlling depth of GHSOM), as reported in Table 4.5 and Table 4.6, respectively.

According to Table 4.5, by varying the threshold τ_1 by 0.1000 starting from 1.0000 to 0.1000 and keeping the threshold $\tau_2 = 0.0035$ unchanged. The results show that setting the threshold τ_1 to 1 would lead to a large number of layers with only 2×2 maps at Layer 1 and setting it to 0 would lead to a small number of layers with a huge map at Layer 1.

Table 4.5 Structure of GHSOMs (by varying τ_1 or breadth)

	$\tau_1 = 1.0000$, $\tau_2 = 0.0035$		$\tau_1 = 0.9000$, $\tau_2 = 0.0035$		$\tau_1 = 0.8000$, $\tau_2 = 0.0035$		$\tau_1 = 0.7000$, $\tau_2 = 0.0035$		$\tau_1 = 0.6000$, $\tau_2 = 0.0035$	
	# L	L1	# L	L1	# L	L1	# L	L1	# L	L1
Data Set 1	6	2×2	5	2×3	4	4×5	4	6×6	3	7×9
Data Set 2	6	2×2	4	4×5	4	10×5	4	12×8	3	13×11
Data Set 3	6	2×2	5	4×3	5	6×4	3	9×5	3	12×8
Data Set 4	6	2×2	4	5×3	4	6×6	3	10×8	3	13×11
Data Set 5	6	2×2	5	6×3	4	12×5	3	16×8	3	16×8
	$\tau_1 = 0.5000$, $\tau_2 = 0.0035$		$\tau_1 = 0.4000$, $\tau_2 = 0.0035$		$\tau_1 = 0.3000$, $\tau_2 = 0.0035$		$\tau_1 = 0.2000$, $\tau_2 = 0.0035$		$\tau_1 = 0.1000$, $\tau_2 = 0.0035$	
	# L	L1	# L	L1	# L	L1	# L	L1	# L	L1
Data Set 1	3	10×10	3	11×13	2	11×13	2	11×13	2	11×13
Data Set 2	3	13×11	3	13×11	2	13×11	2	13×11	2	13×11
Data Set 3	3	13×11	2	13×11	2	13×11	2	13×11	2	13×11
Data Set 4	3	13×11	3	13×11	2	13×11	2	13×11	2	13×11
Data Set 5	3	16×8	2	16×8	2	16×8	2	16×8	2	16×8

According to Table 4.6, by fixing the threshold $\tau_1 = 0.8000$ and varying the threshold τ_2 by half of the previous value starting from 0.8000 to 0.0016. The results show that setting the threshold τ_2 to 1 would lead to no hierarchy and setting it to 0 would lead to very deep branches, while the map size at L1 is stable.

Table 4.6 Structure of GHSOMs (by varying τ_2 or depth)

	$\tau_1 = 0.8000$, $\tau_2 = 0.8000$		$\tau_1 = 0.8000$, $\tau_2 = 0.4000$		$\tau_1 = 0.8000$, $\tau_2 = 0.2000$		$\tau_1 = 0.8000$, $\tau_2 = 0.1000$		$\tau_1 = 0.8000$, $\tau_2 = 0.0500$	
	# L	L1	# L	L1	# L	L1	# L	L1	# L	L1
Data Set 1	1	4×5	1	4×5	1	4×5	1	4×5	2	4×5
Data Set 2	1	10×5	1	10×5	1	10×5	1	10×5	2	10×5
Data Set 3	1	6×4	1	6×4	1	6×4	2	6×4	2	6×4
Data Set 4	1	6×6	1	6×6	1	6×6	1	6×6	2	6×6
Data Set 5	1	12×5	1	12×5	1	12×5	1	12×5	1	12×5
	$\tau_1 = 0.8000$, $\tau_2 = 0.0250$		$\tau_1 = 0.8000$, $\tau_2 = 0.0125$		$\tau_1 = 0.8000$, $\tau_2 = 0.0063$		$\tau_1 = 0.8000$, $\tau_2 = 0.0032$		$\tau_1 = 0.8000$, $\tau_2 = 0.0016$	
	# L	L1	# L	L1	# L	L1	# L	L1	# L	L1
Data Set 1	3	4×5	3	4×5	4	4×5	4	4×5	4	4×5
Data Set 2	2	10×5	3	10×5	3	10×5	4	10×5	4	10×5
Data Set 3	3	6×4	4	6×4	4	6×4	5	6×4	6	6×4
Data Set 4	3	6×6	3	6×6	3	6×6	4	6×6	5	6×6
Data Set 5	2	12×5	3	12×5	3	12×5	4	12×5	4	12×5

4.4.2.3 Training Time

Time spent on training SOMs and GHSOMs for each data set were analyzed. Suppose that the experiments were carried out in a virtual control environment, where only the MATLAB application with SOM Toolbox and GHSOM Toolbox run on the computer system.

- Training Time of SOMs

Table 4.7 reports time spent on training fixed size SOMs with different map size including 10×10, 15×15, 20×20, 25×25, and 30×30. Table 4.8 reports time spent on training recommended size SOMs with additional number of rows and columns, i.e., with row+5 and col+5, with row+10 and col+10, and with double row and double column.

Graphs in Figure 4.25 and 4.26 show the training time of fixed size SOMs and recommended size SOMs, respectively. The results show that for every data set, when the map size of SOM increases, time required for training SOMs trend to increase. It is

important to note that in Figure 4.26 although the recommended size SOM has smaller map size than the one with row+5 and col+5, it requires more time because the SOM Toolbox needs some times to determine the map size.

Table 4.7 Training time (in seconds) of fixed size SOMs

	SOM 10×10	SOM 15×15	SOM 20×20	SOM 25×25	SOM 30×30
Data Set 1	9s	18s	49s	132s	342s
Data Set 2	23s	35s	73s	177s	418s
Data Set 3	11s	14s	25s	53s	117s
Data Set 4	35s	48s	91s	203s	455s
Data Set 5	42s	52s	82s	157s	317s

Table 4.8 Training time (in seconds) of recommended size SOMs

	Recommended Size		With Row+5, Col+5		With Row+10, Col+10		With Row*2, Col*2	
	size	time	size	time	size	time	size	time
Data Set 1	10×8	15s	15×13	15s	20×18	40s	20×16	31s
Data Set 2	13×7	41s	18×12	33s	23×17	71s	26×14	63s
Data Set 3	14×11	16s	19×16	18s	24×21	39s	28×22	51s
Data Set 4	13×8	65s	18×13	50s	23×18	96s	26×16	97s
Data Set 5	14×10	102s	19×15	60s	24×20	105s	28×20	132s

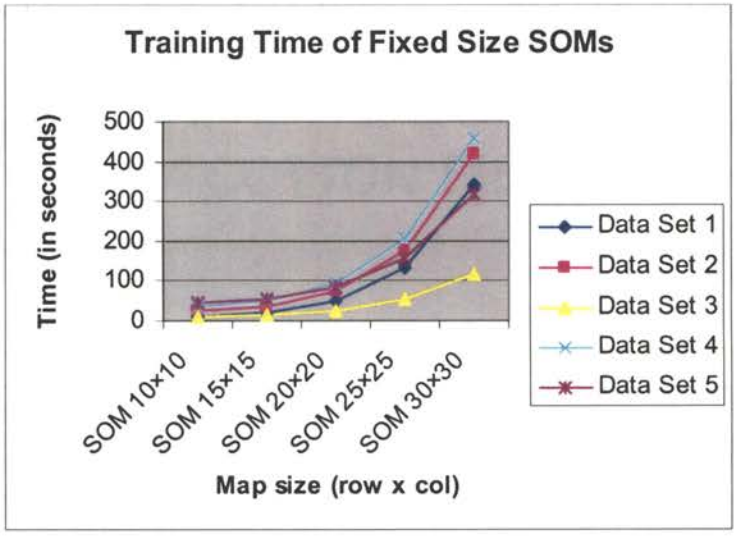


Figure 4.25 Training time of fixed size SOMs

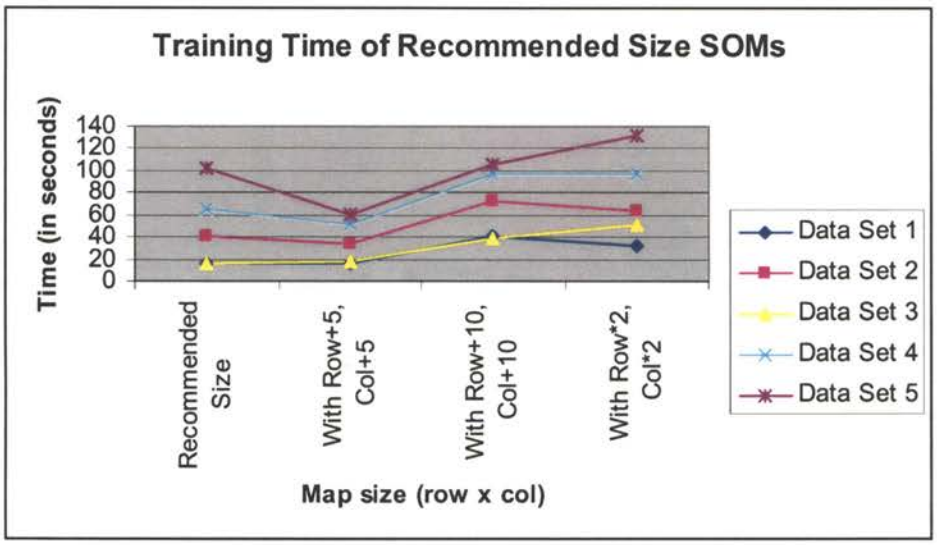


Figure 4.26 Training time of recommended size SOMs

We observed that the graph in Figure 4.25 increases quite rapidly in an apparently exponential manner. It may be desirable to transform it into a logarithmic graph so that the growth rate and the trend can be illustrated more clearly.

- Training time of GHSOMs

Table 4.9 report time spent on training GHSOMs (by varying τ_1 or breadth), by varying the threshold τ_1 by 0.1000 starting from 1.0000 to 0.1000 and keeping the threshold $\tau_2 = 0.0035$ unchanged. Table 4.10 report time spent on training GHSOMs (by varying τ_2 or depth), by fixing the threshold $\tau_1 = 0.8000$ and varying the threshold τ_2 by half of the previous value starting from 0.8000 to 0.0016.

Graphs in Figure 4.27 show the training time of GHSOMs (by varying τ_1 or breadth). The results show that for every data set, time required for training GHSOMs trends to increase when the value of the threshold τ_1 decreases. The reason is that decreasing the value of the threshold τ_1 makes the GHSOM a big flat map.

Graphs in 4.28 show the training time of GHSOMs (by varying τ_2 or depth). The results show that for every data set, time required for training GHSOMs trends to increase when the value of the threshold τ_2 decreases. The reason is that decreasing the value of the threshold τ_2 makes the GHSOM a very deep branches map.

Table 4.9 Training time (in seconds) of GHSOMs (by varying τ_1 or breadth)

	$\tau_1 = 1.0000$, $\tau_2 = 0.0035$	$\tau_1 = 0.9000$, $\tau_2 = 0.0035$	$\tau_1 = 0.8000$, $\tau_2 = 0.0035$	$\tau_1 = 0.7000$, $\tau_2 = 0.0035$	$\tau_1 = 0.6000$, $\tau_2 = 0.0035$
Data Set 1	23s	24s	27s	32s	35s
Data Set 2	59s	75s	86s	105s	119s
Data Set 3	35s	44s	55s	71s	98s
Data Set 4	102s	112s	129s	155s	202s
Data Set 5	144s	182s	262s	358s	365s
	$\tau_1 = 0.5000$, $\tau_2 = 0.0035$	$\tau_1 = 0.4000$, $\tau_2 = 0.0035$	$\tau_1 = 0.3000$, $\tau_2 = 0.0035$	$\tau_1 = 0.2000$, $\tau_2 = 0.0035$	$\tau_1 = 0.1000$, $\tau_2 = 0.0035$
Data Set 1	43s	49s	47s	48s	51s
Data Set 2	126s	132s	134s	142s	144s
Data Set 3	120s	124s	125s	134s	140s
Data Set 4	209s	215s	221s	226s	235s
Data Set 5	374s	394s	411s	431s	464s

Table 4.10 Training time (in seconds) of GHSOMs (by varying τ_2 or depth)

	$\tau_1 = 0.8000$, $\tau_2 = 0.8000$	$\tau_1 = 0.8000$, $\tau_2 = 0.4000$	$\tau_1 = 0.8000$, $\tau_2 = 0.2000$	$\tau_1 = 0.8000$, $\tau_2 = 0.1000$	$\tau_1 = 0.8000$, $\tau_2 = 0.0500$
Data Set 1	13s	13s	13s	13s	18s
Data Set 2	52s	51s	51s	51s	56s
Data Set 3	26s	27s	28s	26s	30s
Data Set 4	63s	62s	62s	66s	72s
Data Set 5	137s	138s	137s	139s	140s
	$\tau_1 = 0.8000$, $\tau_2 = 0.0250$	$\tau_1 = 0.8000$, $\tau_2 = 0.0125$	$\tau_1 = 0.8000$, $\tau_2 = 0.0063$	$\tau_1 = 0.8000$, $\tau_2 = 0.0032$	$\tau_1 = 0.8000$, $\tau_2 = 0.0016$
Data Set 1	20s	23s	27s	27s	28s
Data Set 2	68s	78s	85s	87s	89s
Data Set 3	36s	42s	52s	55s	63s
Data Set 4	94s	114s	123s	125s	131s
Data Set 5	159s	225s	256s	264s	286s

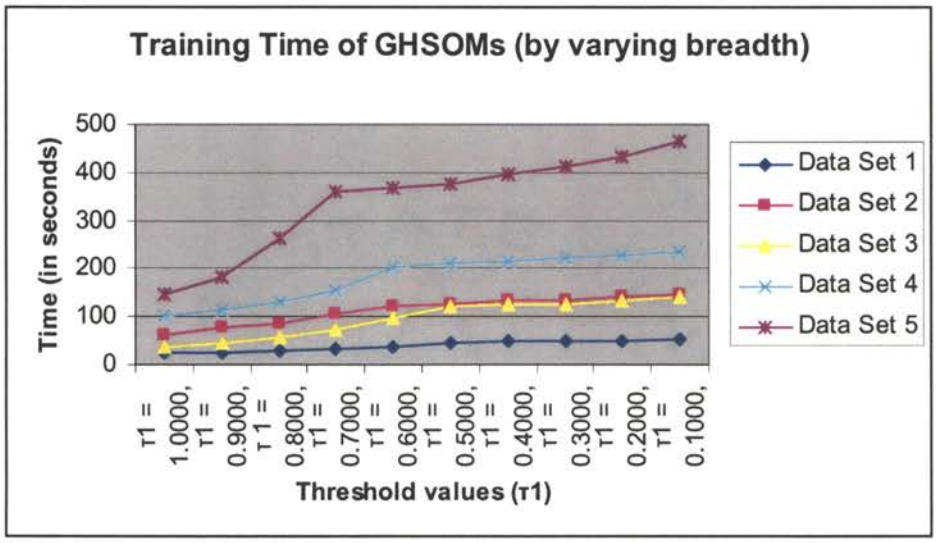


Figure 4.27 Training time of GHSOMs (by varying τ_1 or breadth)

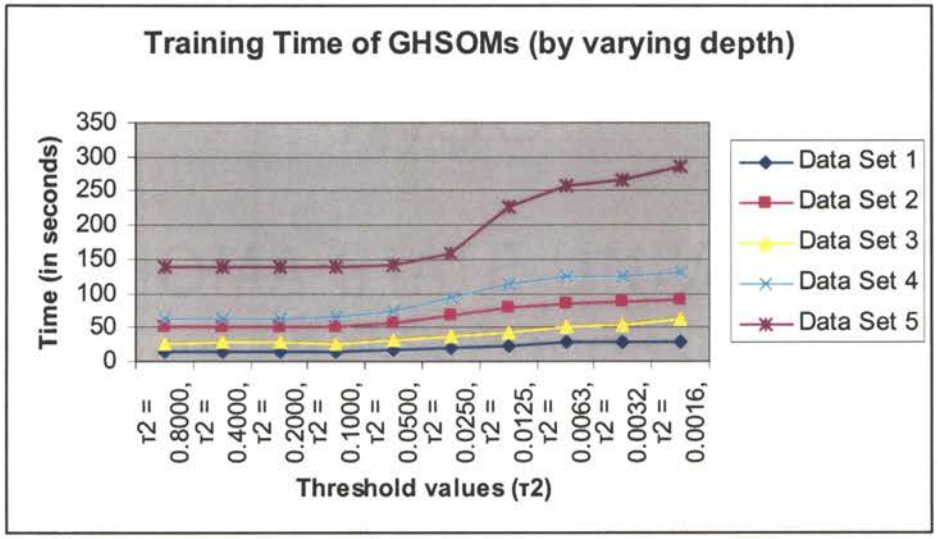


Figure 4.28 Training time of GHSOMs (by varying τ_2 or depth)

4.4.3 Source Code Itemsets for Mining Association Rules

The Understand for C++ tool was utilized to analyze the software components in each data set. It generated a list of all include files that the software components contain. Table 4.11 reports the number of unique source code items or include files extracted from the five data sets.

Table 4.11 Source code itemsets

	Description	No of Files	No of Items
Data Set 1	DS, IR, AI	273	133
Data Set 2	Machine Learning C++	351	205
Data Set 3	GNU Scientific Library	998	423
Data Set 4	AR, SV, GA, FL, NN, DT	413	235
Data Set 5	Data Set 2 and Data Set 4	764	422

A portion of the source code itemsets file for Data Set 1 is given in Figure 4.29. Each line represents a software component. In the third column, itemsets contain lists of all include files, which serve as input transactions for mining association rules. For the complete source code itemsets file for Data Set 1, see Appendix D.

NO.	Code	Include Files
1	DS001	AATree.h iostream.h
2	DS002	AATree.cpp dsexceptions.h iostream.h
3	DS003	AvlTree.h iostream.h
4	DS004	AvlTree.cpp dsexceptions.h iostream.h
5	DS005	BinaryHeap.h
6	DS006	BinaryHeap.cpp dsexceptions.h vector.h
7	DS007	BinarySearchTree.h iostream.h
8	DS008	BinarySearchTree.cpp dsexceptions.h iostream.h
9	DS009	BinomialQueue.h dsexceptions.h
10	DS010	BinomialQueue.cpp iostream.h vector.h

Figure 4.29 A portion of the source code itemsets file for Data Set 1

4.4.4 Interesting Association Rules Discovered

The Apriori program was used to discover a number of interesting association rules from the source code itemsets files. The "interestingness" of the association rules is defined by two measures: support and confidence, as explained in Section 3.4. A number of interesting association rules discovered for several submaps of Data Sets 1 and 4 were examined. They are described below.

- Data Set 1

Referring to the AI submap in Figure 4.4(a), 52 software components were situated in this submap and 36 items of include files were identified. By setting minimum support = 10% and minimum confidence = 10%, 11 association rules were generated. Ten of these association rules are shown in Table 4.12. The first rule means that 23.1% (support) of the software components include "Compare.H". The tenth rule implies that of those software components that include "State.H", 66.7% (confidence) also likely to include "Searches.H".

Table 4.12 Association rules of the AI submap

Association Rules	Support	Confidence
1) -> Compare.H	23.1%	23.1%
2) -> SLBag.H	23.1%	23.1%
3) -> XDString.H	19.2%	19.2%
4) -> State.H	17.3%	17.3%
5) -> Searches.H	17.3%	17.3%
6) Compare.H -> SLBag.H	15.4%	66.7%
7) SLBag.H -> Compare.H	15.4%	66.7%
8) -> SortedQueue.H	13.5%	13.5%
9) -> Queue.H	13.5%	13.5%
10) State.H -> Searches.H	11.5%	66.7%

Considering the DS submap in Figure 4.4(b), 22 software components were mapped onto this submap and 26 items of include files were determined. By setting minimum support = 4% and minimum confidence = 8%, 131 association rules were found. Ten of these association rules are presented in Table 4.13. The fourth rule shows that 31.8% (support) of the software components include "iostream.h", and 46.7% (confidence) are also likely to include "dsexceptions.h". The eighth rule indicates that of the software components under study, 4.5% (support) include "limits.h" and "Random.h", and there is a 100.0% probability (confidence) that "iostream.h" will be included as well.

Table 4.13 Association rules of the DS submap

Association Rules	Support	Confidence
1) -> iostream.h	68.2%	68.2%
2) -> dsexceptions.h	45.5%	45.5%
3) dsexceptions.h -> iostream.h	31.8%	70.0%
4) iostream.h -> dsexceptions.h	31.8%	46.7%
5) -> vector.h	9.1%	9.1%
6) Treap.h -> iostream.h	4.5%	100.0%
7) limits.h Random.h -> dsexceptions.h	4.5%	100.0%
8) limits.h Random.h -> iostream.h	4.5%	100.0%
9) limits.h Random.h Treap.cpp -> iostream.h	4.5%	100.0%
10) limits.h Random.h Treap.cpp dsexceptions.h -> iostream.h	4.5%	100.0%

- Data Set 4

According to the NN submap in Figure 4.16(a), 44 software components were found in this submap and 24 items of include files were identified. By setting minimum support = 10% and minimum confidence = 10%, 90 association rules were produced. Ten

of these association rules are given in Table 4.14. The second rule means that 25.0% (support) of the software components include "graph.h". The ninth rule implies that of the software components under study, 12.5% (support) include "nn_base.h" and "map", and there is a 100.0% probability (confidence) that "graph.h" will be included as well.

Table 4.14 Association rules of the NN submap

Association Rules	Support	Confidence
1) -> object.h	37.5%	37.5%
2) -> graph.h	25.0%	25.0%
3) -> map	20.8%	20.8%
4) -> Math.h	20.8%	20.8%
5) pararr.h -> object.h	16.7%	100.0%
6) string -> map	16.7%	100.0%
7) nn_base.h -> graph.h	16.7%	100.0%
8) pararr.h Math.h -> object.h	12.5%	100.0%
9) nn_base.h map -> graph.h	12.5%	100.0%
10) nn_base.h vector stdexcept -> graph.h	12.5%	100.0%

Considering the FL submap in Figure 4.16(b), 36 software components were mapped onto this submap and 28 items of include files were found. By setting minimum support = 10% and minimum confidence = 10%, 49 association rules were discovered. Ten of these association rules are given in Table 4.15. The seventh rule shows that 13.9% (support) of the software components include "FuzzyModeBase.h", and 71.4% (confidence) are also likely to include "FuzzyVariableBase.h". The ninth rule implies that of the software components under study, 11.1% (support) include "FuzzyVariableBase.h", "FuzzyModeBase.h", and "debug.h", and there is a 80.0% probability (confidence) that "MemberFuncSingle.h" will be included as well.

Table 4.15 Association rules of the FL submap

Association Rules	Support	Confidence
1) -> debug.h	41.7%	41.7%
2) -> FFLLBase.h	25.0%	25.0%
3) -> FuzzyVariableBase.h	25.0%	25.0%
4) FuzzyOutSet.h -> debug.h	19.4%	100.0%
5) FuzzyVariableBase.h -> debug.h	19.4%	77.8%
6) FuzzyModeBase.h debug.h -> FuzzyVariableBase.h	13.9%	83.3%
7) FuzzyModeBase.h -> FuzzyVariableBase.h	13.9%	71.4%
8) FuzzyVariableBase.h -> MemberFuncSingle.h	13.9%	55.6%
9) FuzzyVariableBase.h FuzzyModeBase.h debug.h -> MemberFuncSingle.h	11.1%	80.0%
10) FuzzyModeBase.h -> MemberFuncSingle.h	11.1%	57.1%

CHAPTER V

SUMMARY, CONCLUSIONS, AND FUTURE WORK

5.1 Summary

Chapter I introduces the background information, motivation, and the main objectives of the research. This research work was initiated with an analogy drawn between mining for useful information/knowledge/patterns in a database and searching for reusable components in a software repository. This observation suggests that data mining tools, techniques, and approaches can be practicable in obtaining interesting knowledge from a software repository.

Chapter II provides a survey about the feasibility of applying data mining technology to software reuse, with the goal of discovering useful knowledge from a software repository [Tangsripairoj and Samadzadeh 03-1]. This chapter reviews the general ideas of software reuse and the data mining technology, high-lights several existing data mining applications supporting software reuse, catalogs their distinctive features, and discusses how data mining tools, techniques, and approaches can be applied throughout the process of reuse based software development including acquisition, classification, retrieval, understanding, adaptation, and integration [Tangsripairoj and Samadzadeh 04-1]. Also, Chapter II presents a taxonomy that can be used to categorize data mining applications supporting software reuse [Tangsripairoj and Samadzadeh

03-2]. The taxonomy is based on two major characteristics of the applications: data mining task and data mining technique. The resulting taxonomy provides a predictive framework to help identify possible new data mining applications. In addition, the concepts and methodology of SOM, its application to the organization and visualization of software repositories, and its significant drawbacks are scrutinized and discussed in this chapter [Tangsripairoj and Samadzadeh 04-2].

Chapter III presents the design and methodology of the proposed approach, which is a combination of two effective data mining techniques, namely the GHSOM and the mining association rules. The GHSOM, an improvement over the traditional SOM, is applied to cluster reusable components into groups of semantically similar ones and to facilitate the visualization of the structure of the software repository. Mining association rules are used to discover interesting association rules that represent a number of characteristics of the software components. In this chapter, system architecture of the proposed approach is illustrated and its four major modules: feature extraction, GHSOM construction, mining association rules, and visualization and retrieval, are explained in detail.

Chapter IV describes the experiments and the results, including the experiment objectives, the data sets, and the software tools and computer systems used. The potential of the proposed approach was demonstrated on five data sets consisting of several hundreds of C/C++ program source code files gathered from a number of websites. The results of the GHSOM were compared with the ones obtained by using the traditional SOM with respect to three different perspectives: visualization of the resulting maps, structure of the resulting maps, and training time [Tangsripairoj and Samadzadeh 05].

Additionally, for a particular area of the GHSOM, a number of interesting association rules were discovered and examined.

5.2 Conclusions

We believe that data mining technology is a feasible approach for supporting software reuse. It can be applied to analyze a software repository to look for hidden patterns or possibly unknown relationships among the software components at different phases throughout the process of reuse based software development. The discovered knowledge can help developers to acquire reusable components, organize software repositories, understand the selected components, and find the most suitable components to reuse.

According to the experimental results, we found that the resulting maps of GHSOM, serving as retrieval interfaces, can help developers to obtain better insight into the structure of a software repository, and increase their understanding of the semantic relationships among software components. By using the resulting maps, developers can find the needed software components more easily and quickly, and make better decisions in selecting the best possible components, i.e., optimum "fits", for their needs. The GHSOM is more promising than the traditional SOM owing to its adaptive architecture and the ability to expose the hierarchical structure of data. Moreover, the interesting association rules discovered can be useful in identifying a cohesive set of include files that occur frequently together in a collection of software components.

5.3 Future Work

Some of the possible directions for future work on this research include the following ideas.

First of all, it may be worthwhile to consider using other data mining tasks and techniques. Other analysis tasks of data mining such as regression, summarization, dependency modeling, deviation detection, model visualization, and exploratory data analysis may lead to different kinds of interesting knowledge. Other data mining techniques, e.g., case-based reasoning, Bayesian belief networks, fuzzy sets, genetic algorithms, and rough sets also appear to have a promising potential. For example, the use of GHSOM can be further explored combined with fuzzy sets and genetic algorithms. Fuzzy sets can be helpful when the information about the reusable components is ill-defined. Genetic algorithms can be applied to optimize some threshold values in an attempt to improve performance and obtain better results.

Furthermore, it may be more useful if the software repository to be analyzed contains various kinds of reusable components, e.g., requirement analysis/specifications documents, design patterns, application frameworks, software architectures, database schemas, user interface designs, test plans/cases, software change histories, programmer guides, user manuals, etc.

Also, experiments can be conducted based on larger collections of software components that would represent a truer picture of a real-world software repository.

Another important issue concerns the cluster labeling of the resulting maps, which is currently assigned manually. It may be more desirable to have an automatic generation

scheme for cluster labels in order that developers could locate the needed software components more conveniently and efficiently.

The current study made use of only the include files as source code items for association rules discovery. In fact, other kinds of candidate source code items such as classes, member functions, variables, relationships among various items, and software metrics information, may be used in a supplementary capacity. The result might give developers a more comprehensive view and understanding of the reusable components of interest. For example, the use of coupling and cohesion among software components for determining neighborhoods or closeness relations is considered necessary for more detailed studies. Also, software metrics information used for quantifying software quality is another significant subject matter for further investigation.

REFERENCES

[Agrawal and Srikant 94] Rakesh Agrawal and Ramakrishnan Srikant, "Fast Algorithms for Mining Association Rules", *Proceedings of the 20th International Conference on Very Large Databases,* pp. 487-499, Santiago, Chile, September 1994.

[Agrawal et al. 93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami, "Mining Association Rules Between Sets of Items in Large Databases", *Proceedings of the ACM SIGMOD Conference on Management of Data,* pp. 207-216, Washington, D.C., May 1993.

[Alahakoon et al. 00] Damminda Alahakoon, Saman K. Halgamuge, and Bala Srinivasan, "Dynamic Self-Organizing Maps with Controlled Growth for Knowledge Discovery", *IEEE Transactions on Neural Networks*, Vol. 11, No. 3, pp. 601-614, May 2000.

[Basili et al. 96] Victor R. Basili, Lionel C. Briand, and Walcelio L. Melo, "How Reuse Influences Productivity in Object-Oriented Systems", *Communications of the ACM,* Vol. 39, No. 10, pp. 104-116, October 1996.

[Bauer and Villmann 97] Hans-Ulrich Bauer and Thomas Villmann, "Growing a Hypercubical Output Space in a Self-Organizing Feature Map", *IEEE Transactions on Neural Networks*, Vol. 8, No. 2, pp. 218-2226, March 1997.

[Borgelt 04] Christian Borgelt, "Apriori – Finding Association Rules/Hyperedges with the Apriori Algorithm", http://fuzzy.cs.uni-magdeburg.de/~borgelt/apriori.html, creation date: unknown, last modified date: August 11, 2003, accessed date: October 28, 2004.

[Bennett and Campbell 00] Kristin P. Bennett and Colin Campbell, "Support Vector Machines: Hype or Hallelujah?", *ACM SIGKDD Explorations*, Vol. 2, Issue 2, pp. 1-13, December 2000.

[Blackmore and Miikkulainen 93] Justin Blackmore and Risto Miikkulainen, "Incremental Grid Growing: Encoding High-Dimensional Structure into a Two-Dimensional Feature Map", *Proceedings of the IEEE International Conference on Neural Networks*, Vol. 1, pp. 450-455, San Francisco, California, March-April 1993.

[Brachman et al. 96] Ronald J. Brachman, Tom Khabaza, Willi Kloesgen, Gregory Piatesky-Shapiro, and Evangelos Simoudis, "Mining Business Databases", *Communications of the ACM*, Vol. 39, No. 11, pp. 42-48, November 1996.

[Chan and Pampalk 02] Alvin Chan and Elias Pampalk, "Growing Hierarchical Self Organizing Map (GHSOM) Toolbox: Visualisations and Enhancements", *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'02)*, Vol. 5, pp. 2537-2541, November 2002.

[Chen et al. 96] Ming-Syan Chen, Jiawei Han, and Philip S. Yu, "Data Mining: An Overview from a Database Perspective", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 6, pp. 866-883, December 1996.

[Chi et al. 00] Sheng-Chai Chi, Ren-Jien Kuo, and Po-Wen Teng, "A Fuzzy Self-Organizing Map Neural Network for Market Segmentation of Credit Card", *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 5, pp. 3617-3622, Nashville, Tennessee, October 2000.

[Constantopoulos et al. 95] Panos Constantopoulos, Matthias Jarke, John Mylopoulos, and Yannis Vassiliou, "The Software Information Base: A Server for Reuse", *The International Journal on Very Large Data Bases*, Vol. 4, No. 1, pp. 1-43, January 1995.

[Damiani and Fugini 96] Ernesto Damiani and Maria G. Fugini, "Design and Code Reuse Based on Fuzzy Classification of Components", *ACM SIGAPP Applied Computing Review*, Vol. 4, No. 2, pp. 26-32, Fall 1996.

[Deboeck and Kohonen 98] Guido Deboeck and Teuvo Kohonen (Eds), *Visual Explorations in Finance with Self-Organizing Maps*, Springer, London, UK, 1998.

[Dittenbach et al. 00] Michael Dittenbach, Dieter Merkl, and Andreas Rauber, "The Growing Hierarchical Self-Organizing Map", *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2000)*, pp. 15-19, Como, Italy, July 2000.

[Drobics et al. 01] Mario Drobics, Ulrich Bodenhofer, Werner Winiwater, and Erich Peter Klement, "Data Mining Using Synergies Between Self-Organizing Maps and Inductive Learning of Fuzzy Rules", *Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, Vol. 3, pp. 1780-1785, Vancouver, BC, Canada, July 2001.

[Drummond et al. 00] C. G. Drummond, D. Ionescu, and R. C. Holte, "A Learning Agent that Assists the Browsing of Software Libraries", *IEEE Transactions on Software Engineering*, Vol. 26, No.12, pp. 1179-1196, December 2000.

[El-Khouly et al. 99] M. M. El-Khouly, B. H. Far, and Z. Koono, "A New Multi-Level Information Retrieval Technique for Reuse Software Components", *Proceedings of*

the IEEE International Conference on Systems, Man, and Cybernetics, pp. 773-777, Tokyo, Japan, October 1999.

[Esteva 90] Juan C. Esteva, "Learning to Recognize Reusable Software Modules Using an Inductive Classification System", *Proceedings of the 5th Jerusalem Conference on Information Technology*, Jerusalem, Israel, pp. 278-285, October 1990.

[Fayyad et al. 96] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth, "From Data Mining to Knowledge Discovery: An Overview", *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, Cambridge, MA, 1996.

[Frakes and Fox 95] William B. Frakes and Christopher J. Fox, "Sixteen Questions about Software Reuse", *Communications of the ACM*, Vol. 38, No. 6, pp. 75-87, June 1995.

[Frakes and Isoda 94] William B. Frakes and Sadahiro Isoda, "Success Factors of Systematic Reuse", *IEEE Software*, Vol. 11, Issue 5, pp. 14-19, September 1994.

[Frakes and Pole 94] William B. Frakes and Thomas P. Pole, "An Empirical Study of Representation Methods for Reusable Software Components", *IEEE Transactions on Software Engineering*, Vol. 20, No. 8, pp. 617-630, August 1994.

[Freeman 87] Peter Freeman, "Reusable Software Engineering: Concepts and Research Directions", *Tutorail: Software Reusability*, the IEEE Computer Society, Washington, D.C., 1987.

[Fritzke 94] Bernd Fritzke, "Growing Cell Structures – A Self-Organizing Network for Unsupervised and Supervised Learning", *Neural Networks*, Vol. 7, No. 9, pp. 1441-1460, 1994.

[Fritzke 95] Bernd Fritzke, "Growing Grid – A Self-Organizing Network with Constant Neighborhood Range and Adaptation Strength", *Neural Processing Letters*, Vol. 2, No. 5, pp. 9-13, 1995.

[Ganti et al. 99] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan, "Mining Very Large Databases", *IEEE Computer*, Vol. 32, No. 8, pp. 38-45, August 1999.

[Goebel and Gruenwald 99] Michael Goebel and Le Gruenwald, "A Survey of Data Mining and Knowledge Discovery Software Tools", *ACM SIGKDD*, Vol. 1, No. 1, pp. 20-33, June 1999.

[Groth 98] Robert Groth, *Data Mining: A Hands-on Approach for Business Professionals*, Prentice Hall Publishing Company, Upper Saddle River, New Jersy, 1998.

[Guo and Luqi 00] J. Guo and Luqi, "A Survey of Software Reuse Repositories", *Proceedings of the 7th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, pp. 92-100, Edinburgh, UK, April 2000.

[Ha et al. 99] Seong Wook Ha, Dae-Seong Kang, Kee-Hang Kwan, and Daijin Kim, "n-Rule Genetic Self-Organizing Map Using Genetic Algorithm", *Proceedings of the IEEE International Fuzzy Systems Conference*, Vol. 3, pp. 1781-1784, Seoul, South Korea, August 1999.

[Hall 93] Robert J. Hall "Generalized Behaviror-Based Retrieval", *Proceedings of the 15th International Conference on Software Engineering (ICSE 93)*, pp. 371-380, Baltimore, MD, May 1993.

[Henninger 94] Scott Henninger, "Using Iterative Refinement to Find Reusable Software", *IEEE Software*, Vol. 11, No. 5, pp. 48-59, September 1994.

[Hoglund et al. 00] Albert J. Hoglund, Kimmo Hatonen, and Antti S. Sorvari, "A Computer Host-Based User Anomaly Detection System Using the Self-Organizing Map", *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, Vol. 5, pp. 411-416, Como, Italy, July 2000.

[Houhamdi and Ghoul 01] Z. Houhamdi and S. Ghoul, "A Reuse Description Formalism", *ACS/IEEE International Conference on Computer Systems and Applications*, pp. 395-401, Beirut, Lebanon, June 2001.

[Jain et al. 99] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering: A Review", *ACM Computing Surveys*, Vol. 31, No. 3, pp. 264-323, September 1999.

[Jin et al. 03] Hui-Dong Jin, Kwong-Sak Leung, Man-Leung Wong, and Zong-Ben Xu, "An Efficient Self-Organizing Map Designed by Genetic Algorithms for the Traveling Salesman Problem", *IEEE Transactions on Systems, Man, and Cybernetics*, Part B, Vol. 33, No. 6, pp. 877-888, December 2003.

[Kirk and Zurada 01] James S. Kirk and Jacek M. Zurada, "An Evolutionary Method of Training Topography-Preserving Maps", *Proceedings of the International Joint Conference on Neural Networks*, Vol. 3, pp. 2230-2234, Washington, D.C., July 2001.

[Kleissner 98] Charly Kleissner, "Data Mining for the Enterprise", *Proceedings of the 31st Annual Hawaii International Conference on System Science (HICSS-31)*, pp. 295-304, Kohala Coast, HI, January 1998.

[Kohavi et al. 96] Ron Kohavi, Dan Sommerfield, and James Dougherty, "Data Mining Using MLC++ A Machine Learning Library in C++", *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence,* pp. 234-245, Mountain View, CA, November 1996.

[Kohonen et al. 00] Teuvo Kohonen, Samuel Kaski, Krista Lagus, Jarkko Salojarvi, Jukka Honkela, Vesa Paatero, and Antti Saarela, "*Self Organization of a Massive Document Collection*", *IEEE Transactions on Neural Networks*, Vol. 11, No. 3, pp. 574-585, May 2000.

[Kohonen 01] Teuvo Kohonen, *Self-Organizing Maps*, 3rd Edition, Springer, New York, NY, 2001.

[Krueger 92] Charles W. Krueger, "Software Reuse", *ACM Computing Surveys*, Vol. 24, No. 2, pp. 131-183, June 1992.

[Langley and Simon 95] Pat Langley and Herbert A. Simon, "Applications of Machine Learning and Rule Induction", *Communications of the ACM*, Vol. 38, No. 11, pp. 55-64, November 1995.

[Lee et al. 98] Byung-Jeong Lee, Byung-Ro Moon, and Chi-Su Wu, "Optimization of Multi-way Clustering and Retrieval Using Genetic Algorithms in Reusable Class Library", *Proceedings of the Asia Pacific Software Engineering Conference*, pp. 4-11, Taipei, Taiwan, December 1998.

[Liao et al. 97] Hsian-Chou Liao, Ming-Feng Chen, Feng-Jian Wang, and Jian-Cheng Dai, "Using a Hierarchical Thesaurus for Classifying and Searching Software Libraries", *Proceedings of the 21st Annual International Computer Software and Applications Conference (COMSAC'21)*, pp. 210-216, Washington, D.C., August 1997.

[Luqi and Guo 99] Luqi and J. Guo, "Toward Automated Retrieval for a Software Component Repository", *Proceedings of the IEEE Conference and Workshop on Engineering of Computer-Based Systems*, pp. 99-105, Nashville, Tennessee, March 1999.

[Maarek et al. 91] Yoelle S. Maarek, Daniel M. Berry, and Gail E. Kaiser, "An Information Retrieval Approach For Automatically Constructing Software Libraries", *IEEE Transactions on Software Engineering*, Vol. 17, No. 8, pp. 800-813, August 1991.

[Merkl and Rauber 00] Dieter Merkl and Andreas Rauber, "Digital Libraries – Classification and Visualization Techniques", *Proceedings of the International Conference on Digital Libraries: Research and Practice*, pp. 434-438, Kyoto, Japan, November 2000.

[Merkl et al. 94] Dieter Merkl, A Min Tjoa, and Gerti Kappel, "Learning the Semantic Similarity of Reusable Software Components", *Proceedings of the 3rd International Conference on Software Reuse: Advances in Software Reusability*, pp. 33-41, Rio de Janeiro, Brazil, November 1994.

[Michail 00] Amir Michail, "Data Mining Library Reuse Patterns Using Generalized Association Rules", *Proceedings of the International Conference on Software Engineering (ICSE 2000)*, pp. 167-176, Limerick, Ireland, June 2000.

[Miikkulainen 90] R. Miikkulainen, "Script Recognition with Hierarchical Feature Maps", *Connection Science*, Vol. 2, pp. 83-101, 1990.

[Mili et al. 98] A. Mili, R. Mili, and R. T. Mittermeir, "A Survey of Software Reuse Libraries", *Annals of Software Engineering*, Vol. 5, pp. 349-414, 1998.

[Mili et al. 99] Ali Mili, Sherif Yacoub, Edward Addy, and Hafedh Mili, "Toward an Engineering Discipline of Software Reuse", *IEEE Software*, Vol. 16, No. 5, pp. 22-31, September-October 1999.

[Mitchell 97] Tom M. Mitchell, *Machine Learning*, McGraw-Hill, New York, NY, 1997.

[Mitchell 99] Tom M. Mitchell, "Machine Learning and Data Mining", *Communications of the ACM*, Vol. 42, No. 11, pp. 30-36, November 1999.

[Morisio et al. 02] Maurizio Morisio, Michel Ezran, and Colin Tully, "Success and Failure Factors in Software Reuse", *IEEE Transaction on Software Engineering*, Vol. 28, No. 4, pp. 340-357, April 2002.

[Naenna et al. 03] Thanakorn Naenna, Robert A. Bress, and Mark J. Embrechts, "DNA Classifications with Self-Organizing Maps (SOMs)", *Proceedings of the 2003 IEEE International Workshop on Soft Computing in Industrial Applications*, pp. 151-154, Binghamton, NY, June 2003.

[Pedrycz et al. 01] W. Pedrycz, G. Succi, M. Reformat, P. Musilek, and X. Bai, "Self Organizing Map as a Tool for Software Analysis", *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, pp. 93-97, Toronto, Ontario, Canada, May 2001.

[Porter 80] M.F. Porter, "An Algorithm for Suffix Stripping", *Program*, Vol. 14, No. 3, pp. 30-137, 1980.

[Poulin and Yglesias 93] Jeffrey S. Poulin and Kathryn P. Yglesias, "Experiences with a Faceted Classification Scheme in a Large Reusable Software Library (RSL)", *Proceedings of the 17th IEEE International Computer Software and Applications Conference (COMPSAC* 93), pp. 90-99, Phoenix, Arizona, November 1993.

[Prieto-Diaz 91] Ruben Prieto-Diaz, "Implementing Faceted Classification for Software Reuse", *Communications of the ACM*, Vol. 34, No. 5, pp. 88-97, May 1991.

[Prieto-Diaz 93] Ruben Prieto-Diaz, "Status Report: Software Reusability", *IEEE Software*, Vol. 10, No. 3, pp. 61-66, May 1993.

[Rauber et al. 02] Andreas Rauber, Dieter Merkl, and Michael Dittenbach, "The Growing Hierarchical Self-Organizing Map: Exploratory Analysis of High-Dimensional Data", *IEEE Transactions on Neural Networks*, Vol. 13, No. 6, pp. 1331-1341, November 2002.

[Ravichandran and Rothenberger 03] T. Ravichandran and Marcus A. Rothenberger, "Software Reuse Strategies and Component Markets", *Communications of the ACM*, Vol. 46, No. 8, pp. 109-114, August 2003.

[Reljin et al. 02] Irini S. Reljin, Branimir D. Reljin, and Gordana Jovanovi, "Clustering of Climate Data in Yugoslavia by Using the SOM Neural Network", *Proceedings of the 6th Seminar on Neural Network Applications in Electrical Engineering*, pp. 203-206, Belgrade, Yugoslavia, September 2002.

[Salton 89] Gerard Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[Samadzadeh and Zand 99] M. H. Samadzadeh and M. K. Zand, "Software Houses", *Encyclopedia of Electrical and Electronics Engineering*, Edited by: John G. Webster, John Wiley & Sons, Inc. New York, NY, Vol. 19, pp. 473-483, 1999.

[Smith et al. 98] E. Smith, A. Al-Yasiri, and M. Merabti, "A Multi-Tiered Classification Scheme for Component Retrieval", *Proceedings of the 24th Euromicro Conference*, pp. 882-889, Vesteras, Sweden, August 1998.

[Sommerville 04] Ian Sommerville, *Software Engineering*, 7th Edition, Addison-Wesley Publishing Company, Reading, MA, 2004.

[Sum and Chan 94] John Sum and Lai-Wan Chan, "Fuzzy Self-Organizing Map: Mechanism and Convergence", *Proceedings of the IEEE International Conference on Neural Networks*, Vol. 3, pp. 1674-1679, Orlando, Florida, June 27-July 2, 1994.

[Tanaha et al. 96] Masahiro Tanaha, Yasuyuki Furukawa, and Tetsuzo Tanino, "Weight Tuning and Pattern Classification by Self-Organizing Map Using Genetic Algorithm", *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 602-605, Nagoya, Japan, May 1996.

[Tangsripairoj and Samadzadeh 03-1] Songsri Tangsripairoj and Mansur H. Samadzadeh, "A Survey of Data Mining Technology Applied to Software Reuse", *Proceedings of the 2003 International Conference on Software Engineering Research and Practice (SERP'03)*, pp. 847-853, part of *the 2003 International Multiconference in*

Computer Science and Computer Engineering (15 joint International Conferences), Las Vegas, Nevada, June 2003.

[Tangsripairoj and Samadzadeh 03-2] Songsri Tangsripairoj and Mansur H. Samadzadeh, "A Taxonomy of Data Mining Applications Supporting Software Reuse", Advances in Soft Computing, Edited by: Ajith Abraham, Katrin Franke, and Mario Koppen, pp. 303-312, Springer-Verlag, Heidelberg, Germany, 2003. (This is in fact the edited version of: *Proceedings of the Third International Conference on Intelligent Systems Design and Applications (ISDA '03)*, Tulsa, Oklahoma, August 2003.)

[Tangsripairoj and Samadzadeh 04-1] Songsri Tangsripairoj and Mansur H. Samadzadeh, "Application of Self-Organizing Maps to Software Repositories in Reuse-Based Software Development", *Proceedings of the 2004 International Conference on Software Engineering Research and Practice (SERP '04)*, pp.741-747, part of *the 2004 International Multiconference in Computer Science and Computer Engineering (18 joint International Conferences)*, Las Vegas, Nevada, June 2004.

[Tangsripairoj and Samadzadeh 04-2] Songsri Tangsripairoj and Mansur H. Samadzadeh, "Data Mining Techniques Applied Throughout Reuse Based Software Development", *Proceedings of the 8th World Multi-Conference on Systemics, Cybernetics, and Informatics (SCI '04)*, pp. 480-485, Orlando, FL, July 2004.

[Tangsripairoj and Samadzadeh 05] Songsri Tangsripairoj and Mansur H. Samadzadeh, "Organizing and Visualizing Software Repositories Using the Growing Hierarchical Self-Organizing Map", to appear in the *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC'05)*, Special Track on Software Engineering, Santa Fe, New Mexico, March 2005.

[Tenhagen et al. 01] Andreas Tenhagen, Ulrich Sprekelmeyer, and Wolfram-M. Lippe, "On the Combination of Fuzzy Logic and Kohonen Nets", *Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, Vol. 4, pp. 2144-2149, Vancouver, BC, Canada, July 2001.

[Ugurel et al. 02] Secil Ugurel, Robert Krovetz, C. Lee Giles, David M. Pennock, Eric J. Glover, and Hongyuan Zha, "What's the Code? Automatic Classification of Source Code Archives", *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 632-638, Edmonton, Alberta, Canada, July 2002.

[Vesanto et al. 00] Juha Vesanto, Johan Himberg, Esa Alhoniemi, and Juha Parhankangas, "SOM Toolbox for Matlab 5", *Technical Report A57*, Helsinki University of Technology, http://www.cis.hut.fi/projects/somtoolbox/, April 2000.

[Vuorimaa 94] Petri Vuorimaa, "Use of the Fuzzy Self-Organizing Map in Pattern Recognition", *Proceedings of the 3rd IEEE Conference on Fuzzy Systems*, pp. 798-801, Orlando, Florida, June 1994.

[Ye and Lo 01] H. Ye and B. W. N. Lo, "Towards a Self-Structuring Software Library", *IEE Proceedings: Software*, Vol. 148, No. 2, pp. 45-55, April 2001.

[Zand and Samadzadeh 94] Mansour K. Zand and Mansur H. Samadzadeh, Software Reuse: Issues and Perspectives", *IEEE Potentials*, Vol. 13, Part 3, pp. 15-19, August-September 1994.

APPENDICES

APPENDIX A

GLOSSARY

Apriori Algorithm	An algorithm invented by IBM's Quest project team in 1994 which is used to find association rules from a data set [Agrawal et al. 93].
C4.5	A decision tree algorithm, which is a successor of ID3, and which was developed by J. R. Quinlan in 1993. It has several extended features dealing with unavailable values, continuous attribute value ranges, pruning of decision trees, and rule derivation [Mitchell 97].
CART	Classification And Regression Trees is a decision tree algorithm proposed by Leo Breiman and his colleagues in 1984 [Groth 98]. It produces binary decision trees by using statistical prediction in which each non-leaf node has exactly two branches.
CHAID	Chi-Squared Automatic Interaction Detector is a decision tree algorithm introduced by Gordon B. Kass in 1976 [Groth 98]. It generates decision trees in which the number of branches off of a non-leaf node varies from two to the number of categories of the considered attribute.
CN2	An association rule induction algorithm that was proposed by Peter Clark and Tim Niblett in 1989 [Mitchell 97].
GHSOM	Growing Hierarchical Self-Organizing Map, an extension to the Self-Organizing Map (SOM), is a dynamic SOM model that builds a hierarchy of layers and each layer comprises of independent growing SOMs [Rauber et al. 02].
GSL	The GNU Scientific Library (GSL), a numerical library for C and C++ programmers, is free software under the GNU General Public License. The library provides a wide range of mathematical routines such as random number generators, special functions, statistics, and least-squares fitting.

ID3	Itemized Dichotomizer 3 is a decision tree algorithm created by J. R. Quinlan in 1986 [Groth 98]. It creates decision trees in which the number of branches off of a non-leaf node is equal to the number of categories of the considered attribute.
KDD	Knowledge Discovery in Databases is the process of discovering useful knowledge from a large volume of data [Fayyad et al. 96].
MATLAB	A commercial software developed by MathWorks, Inc. A high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numerical computation.
MLC++	Machine Learning C++ (MLC++) is a library of C++ classes for supervised machine learning which was first developed at Stanford University but is now distributed by Silicon Graphics [Kohavi et al. 96].
NIH	The Not-Invent-Here factor happens because software developers prefer to use their own software components rather than the ones constructed somewhere else [Zand and Samadzadeh 94].
SOFM	Self-Organizing Feature Map is another name for SOM.
SOM	Self-Organizing Map, an unsupervised learning neural network, is a data mining technique for clustering and visualization of huge data sets [Kohonen 01].
SVM	Support Vector Machine is a popular technique for data mining tasks such as classification, regression, and novelty detection [Bennett and Campbell 00].
TF×IDF	The Term Frequency multiplied by Inverse Document Frequency weighting scheme.
U-matrix	The U-matrix or unified distance matrix representation of the SOM helps to visualize distance between neighboring map units, and hence show the cluster structure of the map [Vesanto et al. 00].
VSM	The Vector Space Model [Salton 89], a model that represents each document as a vector of certain weighted word frequencies.

APPENDIX B

LISTS OF FILES IN THE DATA SETS

- Data Set 1: Data Structure (DS), Information Retrieval (IR), and Artificial Intelligence (AI).

NO.	Code	Size (Bytes)	Lines of Comment	Lines of Code	Ratio Comment Per Code	Filename
1	DS001	9733	83	216	0.38	AATree.cpp
2	DS002	2931	18	44	0.41	AATree.h
3	DS003	10757	109	214	0.51	AvlTree.cpp
4	DS004	3247	17	47	0.36	AvlTree.h
5	DS005	3963	51	77	0.66	BinaryHeap.cpp
6	DS006	1548	15	18	0.83	BinaryHeap.h
7	DS007	9213	104	181	0.57	BinarySearchTree.cpp
8	DS008	3002	16	41	0.39	BinarySearchTree.h
9	DS009	9433	72	197	0.37	BinomialQueue.cpp
10	DS010	2734	18	39	0.46	BinomialQueue.h
11	DS011	869	6	27	0.22	BuggyIntCell.cpp
12	DS012	1480	1	35	0.03	Concordance1.cpp
13	DS013	2201	4	64	0.06	Concordance2.cpp
14	DS014	5375	56	121	0.46	CursorList.cpp
15	DS015	3591	27	56	0.48	CursorList.h
16	DS016	6451	54	133	0.41	DSL.cpp
17	DS017	2498	19	37	0.51	DSL.h
18	DS018	1666	25	30	0.83	DisjSets.cpp
19	DS019	976	14	10	1.40	DisjSets.h
20	DS020	306	3	12	0.25	Fig01_02.cpp
21	DS021	313	3	12	0.25	Fig01_03.cpp
22	DS022	435	3	16	0.19	Fig01_04.cpp
23	DS023	1045	17	21	0.81	Fig01_05.cpp
24	DS024	634	10	19	0.53	Fig01_06.cpp
25	DS025	611	2	15	0.13	Fig01_10.cpp
26	DS026	292	6	9	0.67	Fig01_11.cpp
27	DS027	1478	3	53	0.06	Fig01_16.cpp
28	DS028	1106	5	27	0.19	Fig01_19.cpp
29	DS029	1619	13	36	0.36	Fig01_23.cpp
30	DS030	1223	16	27	0.59	Fig02_09.cpp
31	DS031	506	8	16	0.50	Fig02_10.cpp
32	DS032	639	10	21	0.48	Fig02_11.cpp

33	DS033	1148	8	24	0.33	Fig10_38.cpp
34	DS034	1026	10	28	0.36	Fig10_40.cpp
35	DS035	494	2	17	0.12	Fig10_43.cpp
36	DS036	589	2	18	0.11	Fig10_45.cpp
37	DS037	2524	15	46	0.33	Fig10_46.cpp
38	DS038	2517	26	45	0.58	Fig10_53.cpp
39	DS039	1845	19	31	0.61	Fig10_62.cpp
40	DS040	443	1	12	0.08	FigA_04.cpp
41	DS041	1640	0	55	0.00	FigA_05.cpp
42	DS042	644	3	16	0.19	FigA_06.cpp
43	DS043	1264	18	19	0.95	FindMax.cpp
44	DS044	4519	15	151	0.10	Graph1.cpp
45	DS045	5463	17	171	0.10	Graph2.cpp
46	DS046	480	9	11	0.82	IntCell.cpp
47	DS047	380	3	9	0.33	IntCell.h
48	DS048	2699	10	69	0.14	KdTree.cpp
49	DS049	6540	66	136	0.49	LeftistHeap.cpp
50	DS050	2736	16	40	0.40	LeftistHeap.h
51	DS051	3864	48	84	0.57	LinkedList.cpp
52	DS052	3367	30	51	0.59	LinkedList.h
53	DS053	4262	76	93	0.82	MaxSumTest.cpp
54	DS054	661	9	15	0.60	MemoryCell.cpp
55	DS055	543	4	10	0.40	MemoryCell.h
56	DS056	9483	78	183	0.43	PairingHeap.cpp
57	DS057	2593	18	36	0.50	PairingHeap.h
58	DS058	2579	5	70	0.07	Polynomial.cpp
59	DS059	5413	55	115	0.48	QuadraticProbing.cpp
60	DS060	1977	10	31	0.32	QuadraticProbing.h
61	DS061	2453	30	54	0.56	QueueAr.cpp
62	DS062	1458	13	18	0.72	QueueAr.h
63	DS063	1639	19	34	0.56	Random.cpp
64	DS064	993	12	11	1.09	Random.h
65	DS065	10048	85	204	0.42	RedBlackTree.cpp
66	DS066	3624	23	49	0.47	RedBlackTree.h
67	DS067	3564	34	76	0.45	SeparateChaining.cpp
68	DS068	1627	13	18	0.72	SeparateChaining.h
69	DS069	11670	145	218	0.67	Sort.h
70	DS070	10852	92	223	0.41	SplayTree.cpp
71	DS071	2841	17	41	0.41	SplayTree.h
72	DS072	2495	33	48	0.69	StackAr.cpp
73	DS073	1440	14	16	0.88	StackAr.h
74	DS074	3335	39	77	0.51	StackLi.cpp
75	DS075	1754	15	25	0.60	StackLi.h
76	DS076	1470	1	37	0.03	TestAATree.cpp
77	DS077	958	1	22	0.05	TestAvlTree.cpp
78	DS078	949	2	25	0.08	TestBinaryHeap.cpp
79	DS079	1460	1	36	0.03	TestBinarySearchTree.cpp
80	DS080	839	0	25	0.00	TestBinomialQueue.cpp
81	DS081	1308	1	35	0.03	TestCursorList.cpp
82	DS082	911	1	21	0.05	TestDSL.cpp
83	DS083	899	1	24	0.04	TestFastDisjSets.cpp
84	DS084	280	1	7	0.14	TestIntCell.cpp
85	DS085	829	0	25	0.00	TestLeftistHeap.cpp
86	DS086	1337	1	37	0.03	TestLinkedList.cpp
87	DS087	637	2	14	0.14	TestMemoryCell.cpp
88	DS088	1341	1	34	0.03	TestPairingHeap.cpp
89	DS089	893	1	22	0.05	TestQuadraticProbing.cpp

90	DS090	406	0	12	0.00	TestQueueAr.cpp
91	DS091	269	1	7	0.14	TestR2.cpp
92	DS092	267	1	7	0.14	TestRandom.cpp
93	DS093	1278	1	29	0.03	TestRedBlackTree.cpp
94	DS094	894	1	22	0.05	TestSeparateChaining.cpp
95	DS095	2588	31	49	0.63	TestSlowDisjSets.cpp
96	DS096	1626	0	45	0.00	TestSort.cpp
97	DS097	1439	1	36	0.03	TestSplayTree.cpp
98	DS098	314	0	9	0.00	TestStackAr.cpp
99	DS099	499	0	14	0.00	TestStackLi.cpp
100	DS100	494	0	23	0.00	TestString.cpp
101	DS101	1440	1	36	0.03	TestTreap.cpp
102	DS102	8773	78	195	0.40	Treap.cpp
103	DS103	2977	18	44	0.41	Treap.h
104	DS104	98	0	3	0.00	bool.h
105	DS105	201	0	4	0.00	dsexceptions.h
106	DS106	859	0	21	0.00	matrix.h
107	DS107	1943	27	31	0.87	mystring.h
108	DS108	2797	0	107	0.00	string.cpp
109	DS109	866	0	24	0.00	vector.cpp
110	DS110	1308	0	31	0.00	vector.h
111	IR001	8489	209	135	1.55	bool/bv.c
112	IR002	1541	15	19	0.79	bool/bv.h
113	IR003	9560	84	248	0.34	bool/bvdriver.c
114	IR004	10763	87	248	0.35	bool/driver.c
115	IR005	10696	197	207	0.95	bool/hash.c
116	IR006	2327	31	24	1.29	bool/hash.h
117	IR007	10979	94	292	0.32	bool/hdriver.c
118	IR008	1758	33	20	1.65	mphf/comphfns.c
119	IR009	476	11	1	11.00	mphf/comphfns.h
120	IR010	981	21	0	0.00	mphf/const.h
121	IR011	5142	78	73	1.07	mphf/main.c
122	IR012	7000	103	115	0.90	mphf/map.c
123	IR013	6060	75	102	0.74	mphf/order.c
124	IR014	1688	51	22	2.32	mphf/pmrandom.c
125	IR015	840	13	3	4.33	mphf/pmrandom.h
126	IR016	1021	23	12	1.92	mphf/rantab.c
127	IR017	682	12	2	6.00	mphf/rantab.h
128	IR018	2590	42	41	1.02	mphf/regendrv.c
129	IR019	3207	57	45	1.27	mphf/regenphf.c
130	IR020	960	15	10	1.50	mphf/regenphf.h
131	IR021	7422	104	113	0.92	mphf/search.c
132	IR022	4190	65	77	0.84	mphf/support.c
133	IR023	671	9	4	2.25	mphf/support.h
134	IR024	1883	32	27	1.19	mphf/types.h
135	IR025	9842	152	155	0.98	mphf/vheap.c
136	IR026	965	13	5	2.60	mphf/vheap.h
137	IR027	17840	208	235	0.89	stemmer/stem.c
138	IR028	629	10	1	10.00	stemmer/stem.h
139	IR029	3606	62	39	1.59	stemmer/stemmer.c
140	IR030	27127	342	308	1.11	stopper/stop.c
141	IR031	1015	13	3	4.33	stopper/stop.h
142	IR032	1985	37	18	2.06	stopper/stopper.c
143	IR033	15285	271	180	1.51	stopper/strlist.c
144	IR034	2112	27	10	2.70	stopper/strlist.h
145	IR035	1651	7	60	0.12	stringsearch/bm.c
146	IR036	669	5	17	0.29	stringsearch/bmh.c

147	IR037	697	6	17	0.35	stringsearch/bmhs.c
148	IR038	1053	4	42	0.10	stringsearch/kmp.c
149	IR039	1077	12	25	0.48	stringsearch/kr.c
150	IR040	329	2	13	0.15	stringsearch/naive.c
151	IR041	1082	12	25	0.48	stringsearch/rk.c
152	IR042	1001	6	29	0.21	stringsearch/so.c
153	IR043	658	9	0	0.00	stringsearch/string.h
154	IR044	810	7	29	0.24	stringsearch/test.c
155	IR045	30081	428	482	0.89	thesauri/hierarky.c
156	IR046	27392	386	444	0.87	thesauri/merge.c
157	IR047	40633	601	627	0.96	thesauri/select.c
158	AI001	2052	48	0	0.00	03.Logic/C++Code/DataDependencies/DataList.C
159	AI002	10096	85	230	0.37	03.Logic/C++Code/DataDependencies/DataList.H
160	AI003	8658	123	71	1.73	03.Logic/C++Code/DataDependencies/DataNode.C
161	AI004	4032	62	40	1.55	03.Logic/C++Code/DataDependencies/DataNode.H
162	AI005	2779	48	22	2.18	03.Logic/C++Code/DataDependencies/dataTest.C
163	AI006	2132	48	1	48.00	03.Logic/C++Code/DataDependencies/main.H
164	AI007	4748	71	43	1.65	03.Logic/C++Code/Unification/Bind.C
165	AI008	2887	55	14	3.93	03.Logic/C++Code/Unification/Bind.H
166	AI009	426	0	17	0.00	03.Logic/C++Code/Unification/Compare.H
167	AI010	10413	128	136	0.94	03.Logic/C++Code/Unification/LogicNode.C
168	AI011	4607	79	36	2.19	03.Logic/C++Code/Unification/LogicNode.H
169	AI012	11261	134	143	0.94	03.Logic/C++Code/Unification/Parser.C
170	AI013	2411	48	4	12.00	03.Logic/C++Code/Unification/Parser.H
171	AI014	2981	48	26	1.85	03.Logic/C++Code/Unification/testParser.C
172	AI015	4527	64	69	0.93	04.Search/C++Code/Discrimination/DTree.C
173	AI016	2247	53	12	4.42	04.Search/C++Code/Discrimination/DTree.H
174	AI017	3051	53	54	0.98	04.Search/C++Code/Discrimination/Formula.C
175	AI018	2522	56	13	4.31	04.Search/C++Code/Discrimination/Formula.H
176	AI019	3831	67	62	1.08	04.Search/C++Code/Discrimination/Key.C
177	AI020	2667	59	19	3.11	04.Search/C++Code/Discrimination/Key.H
178	AI021	6676	93	99	0.94	04.Search/C++Code/Discrimination/Node.C
179	AI022	3286	67	19	3.53	04.Search/C++Code/Discrimination/Node.H
180	AI023	2231	49	21	2.33	04.Search/C++Code/Discrimination/String.C
181	AI024	1933	47	3	15.67	04.Search/C++Code/Discrimination/String.H
182	AI025	4122	63	70	0.90	04.Search/C++Code/Genetic/Chromosome.C
183	AI026	3023	63	20	3.15	04.Search/C++Code/Genetic/Chromosome.H
184	AI027	9459	99	168	0.59	04.Search/C++Code/Genetic/Population.C
185	AI028	2714	58	17	3.41	04.Search/C++Code/Genetic/Population.H
186	AI029	1458	16	27	0.59	04.Search/C++Code/Search/State.C
187	AI030	1490	11	30	0.37	04.Search/C++Code/Search/State.H
188	AI031	3627	62	22	2.82	04.Search/C++Code/Search/bfs.C
189	AI032	3583	63	24	2.63	04.Search/C++Code/Search/dfs.C
190	AI033	4809	75	42	1.79	04.Search/C++Code/Search/ids.C
191	AI034	5941	81	99	0.82	05.Learning/C++Code/Decision/Decision.C
192	AI035	2759	59	18	3.28	05.Learning/C++Code/Decision/Decision.H
193	AI036	2604	54	25	2.16	05.Learning/C++Code/Decision/Dimension.C
194	AI037	2580	56	15	3.73	05.Learning/C++Code/Decision/Dimension.H
195	AI038	2963	53	34	1.56	05.Learning/C++Code/Decision/Example.C
196	AI039	2449	56	12	4.67	05.Learning/C++Code/Decision/Example.H
197	AI040	10609	120	168	0.71	05.Learning/C++Code/Decision/Node.C
198	AI041	3883	79	27	2.93	05.Learning/C++Code/Decision/Node.H
199	AI042	2231	49	21	2.33	05.Learning/C++Code/Decision/String.C
200	AI043	1933	47	3	15.67	05.Learning/C++Code/Decision/String.H
201	AI044	1971	46	16	2.88	05.Learning/C++Code/PDP/Function.C
202	AI045	1880	46	9	5.11	05.Learning/C++Code/PDP/Function.H
203	AI046	7035	58	172	0.34	05.Learning/C++Code/PDP/PDP.C

204	AI047	2649	48	23	2.09	05.Learning/C++Code/PDP/PDP.H
205	AI048	3361	59	33	1.79	05.Learning/C++Code/PDP/PDP1.C
206	AI049	3323	58	32	1.81	05.Learning/C++Code/PDP/PDP2.C
207	AI050	2472	54	16	3.38	05.Learning/C++Code/Perceptron/Function.C
208	AI051	2395	55	9	6.11	05.Learning/C++Code/Perceptron/Function.H
209	AI052	6600	79	122	0.65	05.Learning/C++Code/Perceptron/Perceptron.C
210	AI053	2231	52	13	4.00	05.Learning/C++Code/Perceptron/Perceptron.H
211	AI054	6013	78	91	0.86	05.Learning/C++Code/Version/Boundary.C
212	AI055	2874	57	16	3.56	05.Learning/C++Code/Version/Boundary.H
213	AI056	5155	72	74	0.97	05.Learning/C++Code/Version/Concept.C
214	AI057	3160	64	23	2.78	05.Learning/C++Code/Version/Concept.H
215	AI058	2469	52	18	2.89	05.Learning/C++Code/Version/Dimension.C
216	AI059	2653	56	14	4.00	05.Learning/C++Code/Version/Dimension.H
217	AI060	2825	48	44	1.09	05.Learning/C++Code/Version/Example.C
218	AI061	2390	52	16	3.25	05.Learning/C++Code/Version/Example.H
219	AI062	2231	49	21	2.33	05.Learning/C++Code/Version/String.C
220	AI063	1933	47	3	15.67	05.Learning/C++Code/Version/String.H
221	AI064	7555	108	101	1.07	05.Learning/C++Code/Version/Version.C
222	AI065	2898	59	20	2.95	05.Learning/C++Code/Version/Version.H
223	AI066	9385	88	141	0.62	6.Advanced/C++Code/Temporal/CausalRuleDatabase.C
224	AI067	6944	89	62	1.44	06.Advanced/C++Code/Temporal/CausalRuleDatabase.H
225	AI068	3212	33	52	0.63	06.Advanced/C++Code/Temporal/Compare.H
226	AI069	15830	200	161	1.24	06.Advanced/C++Code/Temporal/Effects.C
227	AI070	7272	101	88	1.15	06.Advanced/C++Code/Temporal/Effects.H
228	AI071	6941	97	62	1.56	06.Advanced/C++Code/Temporal/FactDatabase.C
229	AI072	8029	93	106	0.88	06.Advanced/C++Code/Temporal/FactDatabase.H
230	AI073	3495	48	40	1.20	06.Advanced/C++Code/Temporal/Project.C
231	AI074	3572	48	35	1.37	06.Advanced/C++Code/Temporal/ScratchNotes.H
232	AI075	2490	48	11	4.36	06.Advanced/C++Code/Temporal/TemporalUpdate.C
233	AI076	2342	48	5	9.60	06.Advanced/C++Code/Temporal/TemporalUpdate.H
234	AI077	3223	54	63	0.86	06.Advanced/C++Code/Temporal/Time.C
235	AI078	3489	66	20	3.30	06.Advanced/C++Code/Temporal/Time.H
236	AI079	1637	0	90	0.00	07.Planning/C++Code/RefinePlans/Compare.H
237	AI080	3944	61	34	1.79	07.Planning/C++Code/RefinePlans/Conflict.C
238	AI081	3598	65	18	3.61	07.Planning/C++Code/RefinePlans/Conflict.H
239	AI082	3486	56	38	1.47	07.Planning/C++Code/RefinePlans/Constrain.C
240	AI083	2855	56	17	3.29	07.Planning/C++Code/RefinePlans/Constrain.H
241	AI084	3461	62	25	2.48	07.Planning/C++Code/RefinePlans/Heuristic.C
242	AI085	2978	59	9	6.56	07.Planning/C++Code/RefinePlans/Heuristic.H
243	AI086	4956	61	69	0.88	07.Planning/C++Code/RefinePlans/Link.C
244	AI087	3700	64	27	2.37	07.Planning/C++Code/RefinePlans/Link.H
245	AI088	5107	80	32	2.50	07.Planning/C++Code/RefinePlans/Operator.C
246	AI089	3052	55	12	4.58	07.Planning/C++Code/RefinePlans/Operator.H
247	AI090	10759	115	139	0.83	07.Planning/C++Code/RefinePlans/Plan.C
248	AI091	5654	77	60	1.28	07.Planning/C++Code/RefinePlans/Plan.H
249	AI092	9470	104	98	1.06	07.Planning/C++Code/RefinePlans/Requirement.C
250	AI093	5061	82	23	3.57	07.Planning/C++Code/RefinePlans/Requirement.H
251	AI094	2434	48	9	5.33	07.Planning/C++Code/RefinePlans/Searches.H
252	AI095	1458	16	27	0.59	07.Planning/C++Code/RefinePlans/State.C
253	AI096	1490	11	30	0.37	07.Planning/C++Code/RefinePlans/State.H
254	AI097	10778	122	133	0.92	07.Planning/C++Code/RefinePlans/Step.C
255	AI098	4281	74	24	3.08	07.Planning/C++Code/RefinePlans/Step.H
256	AI099	4263	72	23	3.13	07.Planning/C++Code/RefinePlans/best.C
257	AI100	6186	48	84	0.57	07.Planning/C++Code/RefinePlans/testPlan.C
258	AI101	516	0	18	0.00	07.Planning/C++Code/StateSpaceSearch/Compare.H
259	AI102	2453	48	16	3.00	07.Planning/C++Code/StateSpaceSearch/Heuristic.C
260	AI103	3043	57	11	5.18	07.Planning/C++Code/StateSpaceSearch/Heuristic.H

261	AI104	5107	80	32	2.50	07.Planning/C++Code/StateSpaceSearch/Operator.C
262	AI105	3052	55	12	4.58	07.Planning/C++Code/StateSpaceSearch/Operator.H
263	AI106	2896	57	5	11.40	07.Planning/C++Code/StateSpaceSearch/Operators.H
264	AI107	6447	92	55	1.67	07.Planning/C++Code/StateSpaceSearch/PlanningState.C
265	AI108	4154	72	16	4.50	07.Planning/C++Code/StateSpaceSearch/PlanningState.H
266	AI109	2434	48	9	5.33	07.Planning/C++Code/StateSpaceSearch/Searches.H
267	AI110	1458	16	27	0.59	07.Planning/C++Code/StateSpaceSearch/State.C
268	AI111	1488	11	30	0.37	07.Planning/C++Code/StateSpaceSearch/State.H
269	AI112	2432	48	10	4.80	07.Planning/C++Code/StateSpaceSearch/StateSearches.C
270	AI113	2304	48	5	9.60	07.Planning/C++Code/StateSpaceSearch/StateSearches.H
271	AI114	4263	72	23	3.13	07.Planning/C++Code/StateSpaceSearch/best.C
272	AI115	3627	62	22	2.82	07.Planning/C++Code/StateSpaceSearch/bfs.C
273	AI116	5210	48	68	0.71	07.Planning/C++Code/StateSpaceSearch/testSSS.C

- Data Set 2: Machine Learning C++ (MLC++).

NO.	Code	Size (Bytes)	Lines of Comment	Lines of Code	Ratio Comment Per Code	Filename
1	MC001	20605	218	243	0.90	src/MCore/Array.c
2	MC002	12819	124	166	0.75	src/MCore/Array2.c
3	MC003	3560	36	57	0.63	src/MCore/BoolArray.c
4	MC004	21582	221	370	0.60	src/MCore/DblLinkList.c
5	MC005	10049	113	126	0.90	src/MCore/DynamicArray.c
6	MC006	7874	97	58	1.67	src/MCore/GenPix.c
7	MC007	8184	90	105	0.86	src/MCore/GetOption.c
8	MC008	12088	115	148	0.78	src/MCore/HashTable.c
9	MC009	5119	48	67	0.72	src/MCore/LogOptions.c
10	MC010	4266	12	131	0.09	src/MCore/MCoreTemplates.c
11	MC011	6843	84	69	1.22	src/MCore/MEnum.c
12	MC012	57382	622	693	0.90	src/MCore/MLCStream.c
13	MC013	20301	193	329	0.59	src/MCore/MOption.c
14	MC014	8828	109	85	1.28	src/MCore/MRandom.c
15	MC015	34555	345	485	0.71	src/MCore/MString.c
16	MC016	2316	28	27	1.04	src/MCore/MaxArray.c
17	MC017	2291	28	27	1.04	src/MCore/MinArray.c
18	MC018	3191	37	34	1.09	src/MCore/RandCharArray.c
19	MC019	12874	139	154	0.90	src/MCore/StatData.c
20	MC020	5663	68	41	1.66	src/MCore/UnivHashTable.c
21	MC021	7852	73	93	0.78	src/MCore/basicCore.c
22	MC022	1383	20	16	1.25	src/MCore/centerline.c
23	MC023	3637	39	48	0.81	src/MCore/checkstream.c
24	MC024	6327	74	55	1.35	src/MCore/error.c
25	MC025	4519	51	42	1.21	src/MCore/fatal_abort.c
26	MC026	1261	14	16	0.88	src/MCore/get_env.c
27	MC027	751	11	5	2.20	src/MCore/machine.c
28	MC028	10851	99	162	0.61	src/MCore/mlcIO.c
29	MC029	14075	198	154	1.29	src/MCore/random.c
30	MC030	2998	42	23	1.83	src/MCore/safe_new.c
31	MC031	2161	60	20	3.00	src/MCore/sortCompare.c
32	MF032	11435	134	117	1.15	src/MFSS/AccEstState.c
33	MF033	8324	53	132	0.40	src/MFSS/BFSearch.c
34	MF034	2233	18	23	0.78	src/MFSS/C45APInducer.c
35	MF035	7309	69	125	0.55	src/MFSS/C45APState.c

36	MF036	7265	69	94	0.73	src/MFSS/CascadeCat.c
37	MF037	7099	58	104	0.56	src/MFSS/CompState.c
38	MF038	8540	67	119	0.56	src/MFSS/DiscSearchInd.c
39	MF039	8866	93	121	0.77	src/MFSS/DiscState.c
40	MF040	4777	46	58	0.79	src/MFSS/FSSInducer.c
41	MF041	4522	50	48	1.04	src/MFSS/FSSState.c
42	MF042	4691	32	72	0.44	src/MFSS/HCSearch.c
43	MF043	999	8	17	0.47	src/MFSS/MFSSTemplates.c
44	MF044	4838	22	89	0.25	src/MFSS/OrderFSSInd.c
45	MF045	1405	21	6	3.50	src/MFSS/OrderFSSState.c
46	MF046	4712	46	58	0.79	src/MFSS/OrderState.c
47	MF047	5843	70	65	1.08	src/MFSS/ProjectCat.c
48	MF048	15342	191	162	1.18	src/MFSS/ProjectInd.c
49	MF049	1571	20	24	0.83	src/MFSS/SANode.c
50	MF050	18944	86	358	0.24	src/MFSS/SASearch.c
51	MF051	3831	37	20	1.85	src/MFSS/SSSearch.c
52	MF052	12983	100	207	0.48	src/MFSS/SearchInducer.c
53	MF053	10059	118	115	1.03	src/MFSS/State.c
54	MF054	13120	105	232	0.45	src/MFSS/StateSpace.c
55	MF055	7550	71	112	0.63	src/MFSS/TableCasInd.c
56	MF056	3151	24	41	0.59	src/MFSS/search_ind.c
57	MF057	1658	16	24	0.67	src/MFSS/sim_anneal.c
58	MF058	6233	50	86	0.58	src/MFSS/WeightSearchInd.c
59	MF059	6002	61	78	0.78	src/MFSS/WeightState.c
60	MG060	17156	172	207	0.83	src/MGLD/Diagram.c
61	MG061	28071	263	346	0.76	src/MGLD/DiagramMngr.c
62	MG062	31822	349	384	0.91	src/MGLD/DisplayMngr.c
63	MG063	8708	89	86	1.03	src/MGLD/GLD.c
64	MG064	11554	124	162	0.77	src/MGLD/GLDPref.c
65	MG065	681	6	10	0.60	src/MGLD/MGLDTemplates.c
66	MG066	28979	199	508	0.39	src/MGLD/Shape.c
67	MH067	9575	95	93	1.02	src/MGraph/DestArray.c
68	MH068	16091	124	283	0.44	src/MGraph/HOODGCIH.c
69	MH069	21107	156	308	0.51	src/MGraph/HOODGInducer.c
70	MH070	660	6	8	0.75	src/MGraph/MGraphTemplates.c
71	MH071	31240	280	433	0.65	src/MGraph/ODGInducer.c
72	MH072	21348	159	317	0.50	src/MGraph/OODGInducer.c
73	MH073	13604	125	189	0.66	src/MGraph/ProjBag.c
74	MH074	3498	30	37	0.81	src/MGraph/ProjStats.c
75	MI075	3292	30	59	0.51	src/MInd/BagAndDistance.c
76	MI076	5778	38	115	0.33	src/MInd/BagMinArray.c
77	MI077	11580	81	189	0.43	src/MInd/CatDTInducer.c
78	MI078	10902	89	160	0.56	src/MInd/IBCategorizer.c
79	MI079	12096	100	206	0.49	src/MInd/IBInducer.c
80	MI080	11264	100	154	0.65	src/MInd/LinDiscr.c
81	MI081	738	6	10	0.60	src/MInd/MIndTemplates.c
82	MI082	21096	215	256	0.84	src/MInd/NaiveBayesCat.c
83	MI083	9278	92	123	0.75	src/MInd/NaiveBayesInd.c
84	MI084	9539	91	125	0.73	src/MInd/PtronInducer.c
85	MI085	5146	60	64	0.94	src/MInd/TableInducer.c
86	MI086	9833	83	156	0.53	src/MInd/WinnowInducer.c
87	MN087	3856	40	45	0.89	src/MInstGen/LIGenFunct.c
88	MN088	13787	151	137	1.10	src/MInstGen/LabInstGen.c
89	MN089	767	6	13	0.46	src/MInstGen/MInstGenTemplates.c
90	ML090	10314	96	161	0.60	src/ML/AccData.c
91	ML091	5643	52	69	0.75	src/ML/AttrCat.c
92	ML092	6448	51	98	0.52	src/ML/AttrEqCat.c

93	ML093	41002	360	602	0.60	src/ML/Attribute.c
94	ML094	3893	41	38	1.08	src/ML/AugCategory.c
95	ML095	4021	47	42	1.12	src/ML/BadCat.c
96	ML096	15101	116	226	0.51	src/ML/BagCounters.c
97	ML097	8774	95	87	1.09	src/ML/BagFeature.c
98	ML098	38090	338	572	0.59	src/ML/BagSet.c
99	ML099	7557	89	78	1.14	src/ML/BaseInducer.c
100	ML100	23387	165	392	0.42	src/ML/CatTestResult.c
101	ML101	5497	52	80	0.65	src/ML/Categorizer.c
102	ML102	3984	44	41	1.07	src/ML/ConstCat.c
103	ML103	3016	37	31	1.19	src/ML/ConstInducer.c
104	ML104	11254	113	153	0.74	src/ML/CtrBag.c
105	ML105	4063	46	56	0.82	src/ML/CtrInducer.c
106	ML106	4794	55	62	0.89	src/ML/CtrInstList.c
107	ML107	5646	60	82	0.73	src/ML/DiscCat.c
108	ML108	6293	65	68	0.96	src/ML/DisplayPref.c
109	ML109	4697	53	49	1.08	src/ML/Inducer.c
110	ML110	21891	220	278	0.79	src/ML/InstList.c
111	ML111	14279	117	257	0.46	src/ML/Instance.c
112	ML112	19239	217	213	1.02	src/ML/InstanceHash.c
113	ML113	2569	11	53	0.21	src/ML/MLTemplates.c
114	ML114	3587	24	67	0.36	src/ML/NullInducer.c
115	ML115	7621	89	92	0.97	src/ML/PartialOrder.c
116	ML116	15858	154	231	0.67	src/ML/Schema.c
117	ML117	6402	74	62	1.19	src/ML/TableCat.c
118	ML118	6199	54	72	0.75	src/ML/ThresholdCat.c
119	ML119	866	11	6	1.83	src/ML/basicML.c
120	ML120	3195	35	35	1.00	src/ML/distance.c
121	ML121	28725	198	475	0.42	src/ML/entropy.c
122	ML122	1128	20	9	2.22	src/ML/stubs.c
123	MT123	8239	48	158	0.30	src/MTrans/AhaIBInducer.c
124	MT124	6434	57	89	0.64	src/MTrans/BinningDisc.c
125	MT125	9290	71	155	0.46	src/MTrans/CN2Inducer.c
126	MT126	15924	136	229	0.59	src/MTrans/DiscDispatch.c
127	MT127	14988	114	236	0.48	src/MTrans/EntropyDisc.c
128	MT128	814	6	11	0.55	src/MTrans/MTransTemplates.c
129	MT129	9885	57	180	0.32	src/MTrans/OC1Inducer.c
130	MT130	17200	209	165	1.27	src/MTrans/OneR.c
131	MT131	5005	39	64	0.61	src/MTrans/OneRInducer.c
132	MT132	9833	51	176	0.29	src/MTrans/PeblsInducer.c
133	MT133	16991	165	233	0.71	src/MTrans/RealDiscretizor.c
134	MT134	22620	143	449	0.32	src/MTrans/convDisplay.c
135	MT135	7289	54	122	0.44	src/MTrans/C45Disc.c
136	MT136	6965	53	110	0.48	src/MTrans/T2Disc.c
137	MR137	15517	90	285	0.32	src/MTree/C45Inducer.c
138	MR138	9723	46	203	0.23	src/MTree/C45RInducer.c
139	MR139	5527	41	99	0.41	src/MTree/C45Tree.c
140	MR140	10764	100	195	0.51	src/MTree/CGraph.c
141	MR141	23653	213	335	0.64	src/MTree/CatGraph.c
142	MR142	3530	47	31	1.52	src/MTree/DTCategorizer.c
143	MR143	8169	61	132	0.46	src/MTree/DecisionTree.c
144	MR144	15746	72	284	0.25	src/MTree/EntropyGainCache.c
145	MR145	11279	84	164	0.51	src/MTree/ID3Inducer.c
146	MR146	10960	66	183	0.36	src/MTree/LazyDTCat.c
147	MR147	7433	48	118	0.41	src/MTree/LazyDTInducer.c
148	MR148	768	6	12	0.50	src/MTree/MTreeTemplates.c
149	MR149	6444	73	77	0.95	src/MTree/RDGCat.c

150	MR150	6056	63	80	0.79	src/MTree/RootCatGraph.c
151	MR151	5151	49	72	0.68	src/MTree/SplitInfoCache.c
152	MR152	18319	167	217	0.77	src/MTree/TDDTInducer.c
153	MR153	41273	230	646	0.36	src/MTree/isocat.c
154	MW154	19487	102	354	0.29	src/MWrapper/AccEstDispatch.c
155	MW155	3765	37	45	0.82	src/MWrapper/AccEstInducer.c
156	MW156	13114	115	167	0.69	src/MWrapper/AccEstimator.c
157	MW157	8643	90	127	0.71	src/MWrapper/AttrOrder.c
158	MW158	9983	72	160	0.45	src/MWrapper/BaggingCat.c
159	MW159	9018	69	139	0.50	src/MWrapper/BaggingInd.c
160	MW160	11248	111	142	0.78	src/MWrapper/Bootstrap.c
161	MW161	4873	33	94	0.35	src/MWrapper/CFInducer.c
162	MW162	13295	78	262	0.30	src/MWrapper/COODGInducer.c
163	MW163	4807	41	57	0.72	src/MWrapper/CVIncremental.c
164	MW164	18458	169	248	0.68	src/MWrapper/CValidator.c
165	MW165	9432	89	132	0.67	src/MWrapper/DFInducer.c
166	MW166	17641	111	300	0.37	src/MWrapper/EntropyODGInducer.c
167	MW167	10348	105	172	0.61	src/MWrapper/FeatureSet.c
168	MW168	4435	35	73	0.48	src/MWrapper/FileNames.c
169	MW169	6624	66	90	0.73	src/MWrapper/HoldOut.c
170	MW170	8518	85	106	0.80	src/MWrapper/LearnCurve.c
171	MW171	4164	47	52	0.90	src/MWrapper/ListHOODGInd.c
172	MW172	8010	63	113	0.56	src/MWrapper/ListODGInducer.c
173	MW173	731	6	9	0.67	src/MWrapper/MWrapperTemplates.c
174	MW174	17037	151	265	0.57	src/MWrapper/ProjGraph.c
175	MW175	19775	190	286	0.66	src/MWrapper/ProjLevel.c
176	MW176	16828	162	243	0.67	src/MWrapper/ProjSet.c
177	MW177	17141	159	223	0.71	src/MWrapper/Projection.c
178	MW178	11222	107	125	0.86	src/MWrapper/StratifiedCV.c
179	MW179	9978	32	176	0.18	src/MWrapper/env_inducer.c
180	IN180	2335	8	40	0.20	inc/AccData.h
181	IN181	3951	20	70	0.29	inc/AccEstDispatch.h
182	IN182	1124	5	19	0.26	inc/AccEstInducer.h
183	IN183	2454	11	44	0.25	inc/AccEstState.h
184	IN184	3019	15	53	0.28	inc/AccEstimator.h
185	IN185	1852	4	33	0.12	inc/AhaIBInducer.h
186	IN186	5067	30	84	0.36	inc/Array.h
187	IN187	2651	10	51	0.20	inc/Array2.h
188	IN188	1397	8	20	0.40	inc/AttrCat.h
189	IN189	1648	9	26	0.35	inc/AttrEqCat.h
190	IN190	1339	3	28	0.11	inc/AttrOrder.h
191	IN191	16821	82	280	0.29	inc/Attribute.h
192	IN192	976	5	17	0.29	inc/AugCategory.h
193	IN193	1194	4	22	0.18	inc/BFSearch.h
194	IN194	1009	5	16	0.31	inc/BadCat.h
195	IN195	1093	4	20	0.20	inc/BagAndDistance.h
196	IN196	2136	15	32	0.47	inc/BagCounters.h
197	IN197	1315	5	21	0.24	inc/BagMinArray.h
198	IN198	7386	34	122	0.28	inc/BagSet.h
199	IN199	1938	8	36	0.22	inc/BaggingCat.h
200	IN200	1916	6	36	0.17	inc/BaggingInd.h
201	IN201	3475	20	45	0.44	inc/BaseInducer.h
202	IN202	1183	5	22	0.23	inc/BinningDisc.h
203	IN203	961	6	15	0.40	inc/BoolArray.h
204	IN204	2135	9	41	0.22	inc/Bootstrap.h
205	IN205	1028	4	13	0.31	inc/C45APInducer.h
206	IN206	1970	8	37	0.22	inc/C45APState.h

207	IN207	2948	17	45	0.38	inc/C45Inducer.h
208	IN208	2213	12	34	0.35	inc/C45RInducer.h
209	IN209	1345	5	18	0.28	inc/CFInducer.h
210	IN210	3507	24	59	0.41	inc/CGraph.h
211	IN211	1215	4	20	0.20	inc/CN2Inducer.h
212	IN212	2391	5	46	0.11	inc/COODGInducer.h
213	IN213	730	4	11	0.36	inc/CVIncremental.h
214	IN214	2331	15	43	0.35	inc/CValidator.h
215	IN215	1379	4	23	0.17	inc/CascadeCat.h
216	IN216	2117	8	36	0.22	inc/CatDTInducer.h
217	IN217	2371	9	44	0.20	inc/CatGraph.h
218	IN218	5066	40	72	0.56	inc/CatTestResult.h
219	IN219	3699	30	36	0.83	inc/Categorizer.h
220	IN220	1874	12	24	0.50	inc/CompState.h
221	IN221	1378	10	17	0.59	inc/ConstCat.h
222	IN222	799	6	11	0.55	inc/ConstInducer.h
223	IN223	1983	10	31	0.32	inc/CtrBag.h
224	IN224	1018	5	17	0.29	inc/CtrInducer.h
225	IN225	1442	8	19	0.42	inc/CtrInstList.h
226	IN226	1551	4	26	0.15	inc/DFInducer.h
227	IN227	704	6	8	0.75	inc/DTCategorizer.h
228	IN228	4145	34	60	0.57	inc/DblLinkList.h
229	IN229	902	5	15	0.33	inc/DecisionTree.h
230	IN230	1286	4	16	0.25	inc/DestArray.h
231	IN231	3635	28	51	0.55	inc/Diagram.h
232	IN232	4313	29	75	0.39	inc/DiagramMngr.h
233	IN233	1268	6	20	0.30	inc/DiscCat.h
234	IN234	1247	5	21	0.24	inc/DiscSearchInd.h
235	IN235	1912	6	32	0.19	inc/DiscState.h
236	IN236	2802	2	81	0.02	inc/DisplayMngr.h
237	IN237	3159	13	65	0.20	inc/DisplayPref.h
238	IN238	2004	14	29	0.48	inc/DynamicArray.h
239	IN239	2484	7	50	0.14	inc/EntropyDisc.h
240	IN240	2145	5	42	0.12	inc/EntropyGainCache.h
241	IN241	1303	4	23	0.17	inc/EntropyODGInducer.h
242	IN242	1153	4	19	0.21	inc/FSSInducer.h
243	IN243	1300	8	21	0.38	inc/FSSState.h
244	IN244	2022	4	42	0.10	inc/FeatureSet.h
245	IN245	818	7	16	0.44	inc/FileNames.h
246	IN246	822	6	12	0.50	inc/GLD.h
247	IN247	3044	9	63	0.14	inc/GLDPref.h
248	IN248	1319	12	23	0.52	inc/GenPix.h
249	IN249	4883	17	68	0.25	inc/GetOption.h
250	IN250	768	4	13	0.31	inc/HCSearch.h
251	IN251	3626	31	62	0.50	inc/HOODGCIH.h
252	IN252	1594	4	25	0.16	inc/HOODGInducer.h
253	IN253	1982	14	33	0.42	inc/HashTable.h
254	IN254	1289	7	24	0.29	inc/HoldOut.h
255	IN255	1938	5	34	0.15	inc/IBCategorizer.h
256	IN256	2348	8	49	0.16	inc/IBInducer.h
257	IN257	1021	4	16	0.25	inc/ID3Inducer.h
258	IN258	607	5	7	0.71	inc/IncrInducer.h
259	IN259	1025	4	21	0.19	inc/MEnum.h
260	IN260	1343	8	19	0.42	inc/Inducer.h
261	IN261	2129	13	30	0.43	inc/InstList.h
262	IN262	4041	25	69	0.36	inc/Instance.h
263	IN263	911	0	28	0.00	inc/InstanceAndDistance.h

264	IN264	4459	34	56	0.61	inc/InstanceHash.h
265	IN265	4932	20	92	0.22	inc/InstanceRC.h
266	IN266	1995	15	42	0.36	inc/LIGenFunct.h
267	IN267	2056	12	32	0.38	inc/LabInstGen.h
268	IN268	3371	6	68	0.09	inc/LazyDTCat.h
269	IN269	2031	5	35	0.14	inc/LazyDTInducer.h
270	IN270	1259	4	26	0.15	inc/LearnCurve.h
271	IN271	1963	4	38	0.11	inc/LinDiscr.h
272	IN272	1031	7	14	0.50	inc/ListHOODGInd.h
273	IN273	1538	8	22	0.36	inc/ListODGInducer.h
274	IN274	3284	14	17	0.82	inc/LogOptions.h
275	IN275	8688	48	172	0.28	inc/MLCStream.h
276	IN276	3434	4	84	0.05	inc/MOption.h
277	IN277	1763	21	18	1.17	inc/MRandom.h
278	IN278	913	4	12	0.33	inc/mlcIO.h
279	IN279	7419	28	152	0.18	inc/MStringRC.h
280	IN280	766	4	16	0.25	inc/MaxArray.h
281	IN281	785	4	17	0.24	inc/MinArray.h
282	IN282	2340	15	37	0.41	inc/NaiveBayesCat.h
283	IN283	1488	9	23	0.39	inc/NaiveBayesInd.h
284	IN284	1112	5	19	0.26	inc/NullInducer.h
285	IN285	1560	8	23	0.35	inc/a.h
286	IN286	2102	4	39	0.10	inc/OC1Inducer.h
287	IN287	4490	16	81	0.20	inc/ODGInducer.h
288	IN288	3903	28	51	0.55	inc/OODGInducer.h
289	IN289	2420	7	52	0.13	inc/OneR.h
290	IN290	608	4	6	0.67	inc/OneRInducer.h
291	IN291	1211	4	21	0.19	inc/OrderFSSInd.h
292	IN292	691	4	8	0.50	inc/OrderFSSState.h
293	IN293	1370	9	20	0.45	inc/PartialOrder.h
294	IN294	1826	5	32	0.16	inc/PeblsInducer.h
295	IN295	4401	28	69	0.41	inc/ProjBag.h
296	IN296	2027	10	35	0.29	inc/ProjGraph.h
297	IN297	3504	22	47	0.47	inc/ProjLevel.h
298	IN298	2203	11	43	0.26	inc/ProjSet.h
299	IN299	862	6	14	0.43	inc/ProjStats.h
300	IN300	1200	4	23	0.17	inc/ProjectCat.h
301	IN301	1955	11	28	0.39	inc/ProjectInd.h
302	IN302	3924	22	61	0.36	inc/Projection.h
303	IN303	1500	6	29	0.21	inc/PtronInducer.h
304	IN304	1475	8	20	0.40	inc/RDGCat.h
305	IN305	812	5	12	0.42	inc/RandCharArray.h
306	IN306	2272	7	42	0.17	inc/RealDiscretizor.h
307	IN307	7411	101	69	1.46	inc/RefCount.h
308	IN308	822	4	14	0.29	inc/RootCatGraph.h
309	IN309	633	5	12	0.42	inc/SANode.h
310	IN310	1966	6	36	0.17	inc/SASearch.h
311	IN311	1687	5	32	0.16	inc/SSSearch.h
312	IN312	9367	29	263	0.11	inc/Shape.h
313	IN313	3033	14	51	0.27	inc/Schema.h
314	IN314	4258	18	84	0.21	inc/SchemaRC.h
315	IN315	2520	11	41	0.27	inc/SearchInducer.h
316	IN316	1214	8	21	0.38	inc/SplitInfo.h
317	IN317	1971	8	44	0.18	inc/SplitInfoCache.h
318	IN318	1619	8	26	0.31	inc/StatData.h
319	IN319	3850	22	64	0.34	inc/State.h
320	IN320	2452	13	39	0.33	inc/StateSpace.h

321	IN321	1137	4	22	0.18	inc/StratifiedCV.h
322	IN322	5029	38	69	0.55	inc/TDDTInducer.h
323	IN323	1270	4	22	0.18	inc/TableCasInd.h
324	IN324	1507	5	23	0.22	inc/TableCat.h
325	IN325	1208	5	19	0.26	inc/TableInducer.h
326	IN326	1560	8	23	0.35	inc/ThresholdCat.h
327	IN327	1043	7	12	0.58	inc/UnivHashTable.h
328	IN328	1469	6	31	0.19	inc/WinnowInducer.h
329	IN329	9042	94	53	1.77	inc/basics.h
330	IN330	515	4	4	1.00	inc/checkstream.h
331	IN331	3713	8	74	0.11	inc/convDisplay.h
332	IN332	544	4	5	0.80	inc/distance.h
333	IN333	4259	23	72	0.32	inc/entropy.h
334	IN334	1148	6	11	0.55	inc/env_inducer.h
335	IN335	1732	12	3	4.00	inc/error.h
336	IN336	1346	16	2	8.00	inc/errorUnless.h
337	IN337	939	0	15	0.00	inc/isocat.h
338	IN338	426	5	0	0.00	inc/machSVR4.h
339	IN339	433	5	0	0.00	inc/machSunOS.h
340	IN340	411	4	2	2.00	inc/machine.h
341	IN341	210	1	6	0.17	inc/random.h
342	IN342	375	4	3	1.33	inc/safe_new.h
343	IN343	137	0	1	0.00	inc/sim_anneal.h
344	IN344	494	4	4	1.00	inc/sortCompare.h
345	IN345	1259	6	20	0.30	inc/C45Disc.h
346	IN346	5098	40	70	0.57	inc/MString.h
347	IN347	1252	6	22	0.27	inc/T2Disc.h
348	IN348	2237	10	42	0.24	inc/DiscDispatch.h
349	IN349	1321	5	18	0.28	inc/OrderState.h
350	IN350	1139	4	17	0.24	inc/WeightSearchInd.h
351	IN351	1518	5	28	0.18	inc/WeightState.h

- Data Set 3: GNU Scientific Library (GSL).

NO.	Code	Size (Bytes)	Lines of Comment	Lines of Code	Ratio Comment Per Code	Filename
1	VE001	2443	25	49	0.51	gsl-1.5/vector/vector_source.c
2	VE002	5126	19	163	0.12	gsl-1.5/vector/init_source.c
3	VE003	2305	18	38	0.47	gsl-1.5/vector/file_source.c
4	VE004	1516	18	26	0.69	gsl-1.5/vector/copy_source.c
5	VE005	2592	18	72	0.25	gsl-1.5/vector/swap_source.c
6	VE006	1208	18	19	0.95	gsl-1.5/vector/prop_source.c
7	VE007	6750	21	177	0.12	gsl-1.5/vector/test_complex_source.c
8	VE008	9425	24	254	0.09	gsl-1.5/vector/test_source.c
9	VE009	1603	18	31	0.58	gsl-1.5/vector/test_io.c
10	VE010	1734	18	34	0.53	gsl-1.5/vector/test_complex_io.c
11	VE011	3630	24	121	0.20	gsl-1.5/vector/minmax_source.c
12	VE012	3198	18	104	0.17	gsl-1.5/vector/oper_source.c
13	VE013	1735	21	30	0.70	gsl-1.5/vector/reim_source.c
14	VE014	2455	18	55	0.33	gsl-1.5/vector/subvector_source.c
15	VE015	2108	18	47	0.38	gsl-1.5/vector/view_source.c
16	VE016	598	1	0	0.00	gsl-1.5/vector/gsl_vector.h
17	VE017	6912	23	91	0.25	gsl-1.5/vector/gsl_vector_char.h
18	VE018	763	1	0	0.00	gsl-1.5/vector/gsl_vector_complex.h

19	VE019	7375	23	99	0.23	gsl-1.5/vector/gsl_vector_complex_double.h
20	VE020	8024	23	99	0.23	gsl-1.5/vector/gsl_vector_complex_float.h
21	VE021	8750	23	99	0.23	gsl-1.5/vector/gsl_vector_complex_long_double.h
22	VE022	6402	23	91	0.25	gsl-1.5/vector/gsl_vector_double.h
23	VE023	7049	23	91	0.25	gsl-1.5/vector/gsl_vector_float.h
24	VE024	6775	23	91	0.25	gsl-1.5/vector/gsl_vector_int.h
25	VE025	6912	23	91	0.25	gsl-1.5/vector/gsl_vector_long.h
26	VE026	7871	23	91	0.25	gsl-1.5/vector/gsl_vector_long_double.h
27	VE027	7049	23	91	0.25	gsl-1.5/vector/gsl_vector_short.h
28	VE028	7209	23	91	0.25	gsl-1.5/vector/gsl_vector_uchar.h
29	VE029	7072	23	91	0.25	gsl-1.5/vector/gsl_vector_uint.h
30	VE030	7209	23	91	0.25	gsl-1.5/vector/gsl_vector_ulong.h
31	VE031	7346	23	91	0.25	gsl-1.5/vector/gsl_vector_ushort.h
32	VE032	1770	0	0	0.00	gsl-1.5/vector/init.c
33	VE033	1823	0	0	0.00	gsl-1.5/vector/file.c
34	VE034	2685	19	1	19.00	gsl-1.5/vector/vector.c
35	VE035	1777	0	0	0.00	gsl-1.5/vector/copy.c
36	VE036	1777	0	0	0.00	gsl-1.5/vector/swap.c
37	VE037	1777	0	0	0.00	gsl-1.5/vector/prop.c
38	VE038	1385	0	0	0.00	gsl-1.5/vector/minmax.c
39	VE039	1363	0	0	0.00	gsl-1.5/vector/oper.c
40	VE040	952	0	0	0.00	gsl-1.5/vector/reim.c
41	VE041	3678	0	0	0.00	gsl-1.5/vector/subvector.c
42	VE042	3538	0	0	0.00	gsl-1.5/vector/view.c
43	VE043	79	0	0	0.00	gsl-1.5/vector/view.h
44	VE044	5072	18	71	0.25	gsl-1.5/vector/test.c
45	VE045	105	0	0	0.00	gsl-1.5/vector/test_static.c
46	MA046	3143	26	70	0.37	gsl-1.5/matrix/matrix_source.c
47	MA047	6021	20	192	0.10	gsl-1.5/matrix/init_source.c
48	MA048	4351	26	111	0.23	gsl-1.5/matrix/file_source.c
49	MA049	3500	18	91	0.20	gsl-1.5/matrix/rowcol_source.c
50	MA050	5083	18	154	0.12	gsl-1.5/matrix/swap_source.c
51	MA051	2499	18	55	0.33	gsl-1.5/matrix/copy_source.c
52	MA052	12072	20	329	0.06	gsl-1.5/matrix/test_complex_source.c
53	MA053	12336	19	364	0.05	gsl-1.5/matrix/test_source.c
54	MA054	4682	24	155	0.15	gsl-1.5/matrix/minmax_source.c
55	MA055	1319	18	22	0.82	gsl-1.5/matrix/prop_source.c
56	MA056	4148	18	142	0.13	gsl-1.5/matrix/oper_source.c
57	MA057	5054	3	180	0.02	gsl-1.5/matrix/getset_source.c
58	MA058	7204	19	143	0.13	gsl-1.5/matrix/view_source.c
59	MA059	2066	18	42	0.43	gsl-1.5/matrix/submatrix_source.c
60	MA060	5711	18	174	0.10	gsl-1.5/matrix/oper_complex_source.c
61	MA061	598	1	0	0.00	gsl-1.5/matrix/gsl_matrix.h
62	MA062	10723	26	148	0.18	gsl-1.5/matrix/gsl_matrix_char.h
63	MA063	10960	26	138	0.19	gsl-1.5/matrix/gsl_matrix_complex_double.h
64	MA064	2030	26	138	0.19	gsl-1.5/matrix/gsl_matrix_complex_float.h
65	MA065	13170	26	138	0.19	gsl-1.5/matrix/gsl_matrix_complex_long_double.h
66	MA066	9898	26	148	0.18	gsl-1.5/matrix/gsl_matrix_double.h
67	MA067	10923	26	148	0.18	gsl-1.5/matrix/gsl_matrix_float.h
68	MA068	10523	26	148	0.18	gsl-1.5/matrix/gsl_matrix_int.h
69	MA069	10723	26	148	0.18	gsl-1.5/matrix/gsl_matrix_long.h
70	MA070	12123	26	148	0.18	gsl-1.5/matrix/gsl_matrix_long_double.h
71	MA071	10923	26	148	0.18	gsl-1.5/matrix/gsl_matrix_short.h
72	MA072	11083	26	148	0.18	gsl-1.5/matrix/gsl_matrix_uchar.h
73	MA073	10883	26	148	0.18	gsl-1.5/matrix/gsl_matrix_uint.h
74	MA074	11083	26	148	0.18	gsl-1.5/matrix/gsl_matrix_ulong.h
75	MA075	11283	26	148	0.18	gsl-1.5/matrix/gsl_matrix_ushort.h

76	MA076	1777	0	0	0.00	gsl-1.5/matrix/init.c
77	MA077	1805	0	0	0.00	gsl-1.5/matrix/matrix.c
78	MA078	1805	0	0	0.00	gsl-1.5/matrix/file.c
79	MA079	3655	0	0	0.00	gsl-1.5/matrix/rowcol.c
80	MA080	1805	0	0	0.00	gsl-1.5/matrix/swap.c
81	MA081	1777	0	0	0.00	gsl-1.5/matrix/copy.c
82	MA082	1385	0	0	0.00	gsl-1.5/matrix/minmax.c
83	MA083	1777	0	0	0.00	gsl-1.5/matrix/prop.c
84	MA084	1840	0	0	0.00	gsl-1.5/matrix/oper.c
85	MA085	1833	0	0	0.00	gsl-1.5/matrix/getset.c
86	MA086	3545	0	0	0.00	gsl-1.5/matrix/view.c
87	MA087	3739	0	0	0.00	gsl-1.5/matrix/submatrix.c
88	MA088	165	0	0	0.00	gsl-1.5/matrix/view.h
89	MA089	4790	18	74	0.24	gsl-1.5/matrix/test.c
90	MA090	106	0	0	0.00	gsl-1.5/matrix/test_static.c
91	PE091	3928	35	91	0.38	gsl-1.5/permutation/permute_source.c
92	PE092	3259	20	32	0.63	gsl-1.5/permutation/gsl_permutation.h
93	PE093	614	1	0	0.00	gsl-1.5/permutation/gsl_permute.h
94	PE094	1403	19	4	4.75	gsl-1.5/permutation/gsl_permute_char.h
95	PE095	1482	19	4	4.75	gsl-1.5/permutation/gsl_permute_complex_double.h
96	PE096	1488	19	4	4.75	gsl-1.5/permutation/gsl_permute_complex_float.h
97	PE097	1536	19	4	4.75	gsl-1.5/permutation/gsl_permute_complex_long_double.h
98	PE098	1405	19	4	4.75	gsl-1.5/permutation/gsl_permute_double.h
99	PE099	1411	19	4	4.75	gsl-1.5/permutation/gsl_permute_float.h
100	PE100	1395	19	4	4.75	gsl-1.5/permutation/gsl_permute_int.h
101	PE101	1403	19	4	4.75	gsl-1.5/permutation/gsl_permute_long.h
102	PE102	1459	19	4	4.75	gsl-1.5/permutation/gsl_permute_long_double.h
103	PE103	1411	19	4	4.75	gsl-1.5/permutation/gsl_permute_short.h
104	PE104	1427	19	4	4.75	gsl-1.5/permutation/gsl_permute_uchar.h
105	PE105	1419	19	4	4.75	gsl-1.5/permutation/gsl_permute_uint.h
106	PE106	1427	19	4	4.75	gsl-1.5/permutation/gsl_permute_ulong.h
107	PE107	1435	19	4	4.75	gsl-1.5/permutation/gsl_permute_ushort.h
108	PE108	733	1	0	0.00	gsl-1.5/permutation/gsl_permute_vector.h
109	PE109	1438	19	4	4.75	gsl-1.5/permutation/gsl_permute_vector_char.h
110	PE110	1500	19	4	4.75	gsl-1.5/permutation/gsl_permute_vector_complex_double.h
111	PE111	1519	19	4	4.75	gsl-1.5/permutation/gsl_permute_vector_complex_float.h
112	PE112	1573	19	4	4.75	gsl-1.5/permutation/gsl_permute_vector_complex_long_double.h
113	PE113	1428	19	4	4.75	gsl-1.5/permutation/gsl_permute_vector_double.h
114	PE114	1447	19	4	4.75	gsl-1.5/permutation/gsl_permute_vector_float.h
115	PE115	1429	19	4	4.75	gsl-1.5/permutation/gsl_permute_vector_int.h
116	PE116	1438	19	4	4.75	gsl-1.5/permutation/gsl_permute_vector_long.h
117	PE117	1501	19	4	4.75	gsl-1.5/permutation/gsl_permute_vector_long_double.h
118	PE118	1447	19	4	4.75	gsl-1.5/permutation/gsl_permute_vector_short.h
119	PE119	1447	19	4	4.75	gsl-1.5/permutation/gsl_permute_vector_uchar.h
120	PE120	1438	19	4	4.75	gsl-1.5/permutation/gsl_permute_vector_uint.h
121	PE121	1447	19	4	4.75	gsl-1.5/permutation/gsl_permute_vector_ulong.h
122	PE122	1456	19	4	4.75	gsl-1.5/permutation/gsl_permute_vector_ushort.h
123	PE123	2171	21	54	0.39	gsl-1.5/permutation/init.c
124	PE124	2430	22	58	0.38	gsl-1.5/permutation/file.c
125	PE125	5549	24	207	0.12	gsl-1.5/permutation/permutation.c
126	PE126	1884	0	0	0.00	gsl-1.5/permutation/permute.c
127	PE127	3587	23	128	0.18	gsl-1.5/permutation/canonical.c
128	PE128	9961	20	175	0.11	gsl-1.5/permutation/test.c
129	CO129	2748	21	26	0.81	gsl-1.5/combination/gsl_combination.h
130	CO130	2734	23	78	0.29	gsl-1.5/combination/init.c
131	CO131	2447	23	58	0.40	gsl-1.5/combination/file.c
132	CO132	3939	27	133	0.20	gsl-1.5/combination/combination.c

133	CO133	6510	23	182	0.13	gsl-1.5/combination/test.c
134	SO134	2123	24	53	0.45	gsl-1.5/sort/sortvec_source.c
135	SO135	2460	25	62	0.40	gsl-1.5/sort/sortvecind_source.c
136	SO136	2968	19	94	0.20	gsl-1.5/sort/subset_source.c
137	SO137	3041	19	94	0.20	gsl-1.5/sort/subsetind_source.c
138	SO138	9251	23	206	0.11	gsl-1.5/sort/test_source.c
139	SO139	4056	21	125	0.17	gsl-1.5/sort/test_heapsort.c
140	SO140	1427	19	5	3.80	gsl-1.5/sort/gsl_heapsort.h
141	SO141	435	1	0	0.00	gsl-1.5/sort/gsl_sort.h
142	SO142	1837	19	8	2.38	gsl-1.5/sort/gsl_sort_char.h
143	SO143	1831	19	8	2.38	gsl-1.5/sort/gsl_sort_double.h
144	SO144	1855	19	8	2.38	gsl-1.5/sort/gsl_sort_float.h
145	SO145	1819	19	8	2.38	gsl-1.5/sort/gsl_sort_int.h
146	SO146	1837	19	8	2.38	gsl-1.5/sort/gsl_sort_long.h
147	SO147	1963	19	8	2.38	gsl-1.5/sort/gsl_sort_long_double.h
148	SO148	1855	19	8	2.38	gsl-1.5/sort/gsl_sort_short.h
149	SO149	1919	19	8	2.38	gsl-1.5/sort/gsl_sort_uchar.h
150	SO150	1901	19	8	2.38	gsl-1.5/sort/gsl_sort_uint.h
151	SO151	1919	19	8	2.38	gsl-1.5/sort/gsl_sort_ulong.h
152	SO152	1937	19	8	2.38	gsl-1.5/sort/gsl_sort_ushort.h
153	SO153	533	1	0	0.00	gsl-1.5/sort/gsl_sort_vector.h
154	SO154	1778	19	8	2.38	gsl-1.5/sort/gsl_sort_vector_char.h
155	SO155	1732	19	8	2.38	gsl-1.5/sort/gsl_sort_vector_double.h
156	SO156	1797	19	8	2.38	gsl-1.5/sort/gsl_sort_vector_float.h
157	SO157	1759	19	8	2.38	gsl-1.5/sort/gsl_sort_vector_int.h
158	SO158	1778	19	8	2.38	gsl-1.5/sort/gsl_sort_vector_long.h
159	SO159	1911	19	8	2.38	gsl-1.5/sort/gsl_sort_vector_long_double.h
160	SO160	1797	19	8	2.38	gsl-1.5/sort/gsl_sort_vector_short.h
161	SO161	1813	19	8	2.38	gsl-1.5/sort/gsl_sort_vector_uchar.h
162	SO162	1794	19	8	2.38	gsl-1.5/sort/gsl_sort_vector_uint.h
163	SO163	1813	19	8	2.38	gsl-1.5/sort/gsl_sort_vector_ulong.h
164	SO164	1832	19	8	2.38	gsl-1.5/sort/gsl_sort_vector_ushort.h
165	SO165	2719	28	64	0.44	gsl-1.5/sort/sort.c
166	SO166	2517	27	52	0.52	gsl-1.5/sort/sortind.c
167	SO167	2154	18	0	0.00	gsl-1.5/sort/sortvec.c
168	SO168	2187	18	0	0.00	gsl-1.5/sort/sortvecind.c
169	SO169	1990	14	0	0.00	gsl-1.5/sort/subset.c
170	SO170	2026	14	0	0.00	gsl-1.5/sort/subsetind.c
171	SO171	3360	19	34	0.56	gsl-1.5/sort/test.c
172	CB172	700	0	44	0.00	gsl-1.5/cblas/tests.c
173	CB173	1008	0	44	0.00	gsl-1.5/cblas/tests.h
174	CB174	964	3	0	0.00	gsl-1.5/cblas/cblas.h
175	CB175	1003	18	13	1.38	gsl-1.5/cblas/source_asum_c.h
176	CB176	965	18	13	1.38	gsl-1.5/cblas/source_asum_r.h
177	CB177	1317	18	18	1.00	gsl-1.5/cblas/source_axpy_c.h
178	CB178	1338	18	26	0.69	gsl-1.5/cblas/source_axpy_r.h
179	CB179	1011	18	11	1.64	gsl-1.5/cblas/source_copy_c.h
180	CB180	956	18	10	1.80	gsl-1.5/cblas/source_copy_r.h
181	CB181	1323	18	19	0.95	gsl-1.5/cblas/source_dot_c.h
182	CB182	998	18	12	1.50	gsl-1.5/cblas/source_dot_r.h
183	CB183	5888	23	134	0.17	gsl-1.5/cblas/source_gbmv_c.h
184	CB184	2718	21	69	0.30	gsl-1.5/cblas/source_gbmv_r.h
185	CB185	5948	21	127	0.17	gsl-1.5/cblas/source_gemm_c.h
186	CB186	3091	21	88	0.24	gsl-1.5/cblas/source_gemm_r.h
187	CB187	5116	23	118	0.19	gsl-1.5/cblas/source_gemv_c.h
188	CB188	2404	21	61	0.34	gsl-1.5/cblas/source_gemv_r.h
189	CB189	1455	18	28	0.64	gsl-1.5/cblas/source_ger.h

190	CB190	2333	18	42	0.43	gsl-1.5/cblas/source_gerc.h
191	CB191	2330	18	42	0.43	gsl-1.5/cblas/source_geru.h
192	CB192	5116	22	109	0.20	gsl-1.5/cblas/source_hbmv.h
193	CB193	8087	27	157	0.17	gsl-1.5/cblas/source_hemm.h
194	CB194	5001	22	105	0.21	gsl-1.5/cblas/source_hemv.h
195	CB195	2758	18	55	0.33	gsl-1.5/cblas/source_her.h
196	CB196	4285	22	77	0.29	gsl-1.5/cblas/source_her2.h
197	CB197	10497	42	198	0.21	gsl-1.5/cblas/source_her2k.h
198	CB198	5138	20	127	0.16	gsl-1.5/cblas/source_herk.h
199	CB199	5011	22	105	0.21	gsl-1.5/cblas/source_hpmv.h
200	CB200	2782	18	55	0.33	gsl-1.5/cblas/source_hpr.h
201	CB201	4315	22	77	0.29	gsl-1.5/cblas/source_hpr2.h
202	CB202	1107	18	18	1.00	gsl-1.5/cblas/source_iamax_c.h
203	CB203	1056	18	17	1.06	gsl-1.5/cblas/source_iamax_r.h
204	CB204	1518	18	33	0.55	gsl-1.5/cblas/source_nrm2_c.h
205	CB205	1290	18	25	0.72	gsl-1.5/cblas/source_nrm2_r.h
206	CB206	1040	18	13	1.38	gsl-1.5/cblas/source_rot.h
207	CB207	1312	18	25	0.72	gsl-1.5/cblas/source_rotg.h
208	CB208	1450	18	35	0.51	gsl-1.5/cblas/source_rotm.h
209	CB209	2869	25	112	0.22	gsl-1.5/cblas/source_rotmg.h
210	CB210	2782	20	70	0.29	gsl-1.5/cblas/source_sbmv.h
211	CB211	1222	18	17	1.06	gsl-1.5/cblas/source_scal_c.h
212	CB212	988	18	13	1.38	gsl-1.5/cblas/source_scal_c_s.h
213	CB213	954	18	12	1.50	gsl-1.5/cblas/source_scal_r.h
214	CB214	2742	20	69	0.29	gsl-1.5/cblas/source_spmv.h
215	CB215	1676	18	34	0.53	gsl-1.5/cblas/source_spr.h
216	CB216	1984	18	44	0.41	gsl-1.5/cblas/source_spr2.h
217	CB217	1133	18	15	1.20	gsl-1.5/cblas/source_swap_c.h
218	CB218	1000	18	12	1.50	gsl-1.5/cblas/source_swap_r.h
219	CB219	8324	23	161	0.14	gsl-1.5/cblas/source_symm_c.h
220	CB220	3499	23	90	0.26	gsl-1.5/cblas/source_symm_r.h
221	CB221	2735	20	67	0.30	gsl-1.5/cblas/source_symv.h
222	CB222	1670	18	34	0.53	gsl-1.5/cblas/source_syr.h
223	CB223	1978	18	44	0.41	gsl-1.5/cblas/source_syr2.h
224	CB224	7382	19	152	0.13	gsl-1.5/cblas/source_syr2k_c.h
225	CB225	3311	19	93	0.20	gsl-1.5/cblas/source_syr2k_r.h
226	CB226	5903	20	128	0.16	gsl-1.5/cblas/source_syrk_c.h
227	CB227	3096	19	91	0.21	gsl-1.5/cblas/source_syrk_r.h
228	CB228	6274	22	133	0.17	gsl-1.5/cblas/source_tbmv_c.h
229	CB229	3497	20	76	0.26	gsl-1.5/cblas/source_tbmv_r.h
230	CB230	6676	23	139	0.17	gsl-1.5/cblas/source_tbsv_c.h
231	CB231	3867	24	90	0.27	gsl-1.5/cblas/source_tbsv_r.h
232	CB232	6005	20	141	0.14	gsl-1.5/cblas/source_tpmv_c.h
233	CB233	3246	20	70	0.29	gsl-1.5/cblas/source_tpmv_r.h
234	CB234	8244	23	179	0.13	gsl-1.5/cblas/source_tpsv_c.h
235	CB235	3807	24	98	0.24	gsl-1.5/cblas/source_tpsv_r.h
236	CB236	12285	29	249	0.12	gsl-1.5/cblas/source_trmm_c.h
237	CB237	5324	26	142	0.18	gsl-1.5/cblas/source_trmm_r.h
238	CB238	5926	20	129	0.16	gsl-1.5/cblas/source_trmv_c.h
239	CB239	3525	20	84	0.24	gsl-1.5/cblas/source_trmv_r.h
240	CB240	14776	29	314	0.09	gsl-1.5/cblas/source_trsm_c.h
241	CB241	6365	26	198	0.13	gsl-1.5/cblas/source_trsm_r.h
242	CB242	8140	23	180	0.13	gsl-1.5/cblas/source_trsv_c.h
243	CB243	3739	24	99	0.24	gsl-1.5/cblas/source_trsv_r.h
244	CB244	430	0	22	0.00	gsl-1.5/cblas/hypot.c
245	CB245	33656	80	466	0.17	gsl-1.5/cblas/gsl_cblas.h
246	CB246	199	0	4	0.00	gsl-1.5/cblas/sasum.c

247	CB247	256	0	5	0.00	gsl-1.5/cblas/saxpy.c
248	CB248	199	0	4	0.00	gsl-1.5/cblas/scasum.c
249	CB249	199	0	4	0.00	gsl-1.5/cblas/scnrm2.c
250	CB250	237	0	5	0.00	gsl-1.5/cblas/scopy.c
251	CB251	319	0	5	0.00	gsl-1.5/cblas/sdot.c
252	CB252	345	0	5	0.00	gsl-1.5/cblas/sdsdot.c
253	CB253	437	0	7	0.00	gsl-1.5/cblas/sgbmv.c
254	CB254	468	0	8	0.00	gsl-1.5/cblas/sgemm.c
255	CB255	409	0	7	0.00	gsl-1.5/cblas/sgemv.c
256	CB256	337	0	6	0.00	gsl-1.5/cblas/sger.c
257	CB257	199	0	4	0.00	gsl-1.5/cblas/snrm2.c
258	CB258	256	0	5	0.00	gsl-1.5/cblas/srot.c
259	CB259	191	0	4	0.00	gsl-1.5/cblas/srotg.c
260	CB260	245	0	5	0.00	gsl-1.5/cblas/srotm.c
261	CB261	212	0	4	0.00	gsl-1.5/cblas/srotmg.c
262	CB262	400	0	7	0.00	gsl-1.5/cblas/ssbmv.c
263	CB263	211	0	4	0.00	gsl-1.5/cblas/sscal.c
264	CB264	360	0	6	0.00	gsl-1.5/cblas/sspmv.c
265	CB265	306	0	6	0.00	gsl-1.5/cblas/sspr.c
266	CB266	343	0	6	0.00	gsl-1.5/cblas/sspr2.c
267	CB267	218	0	4	0.00	gsl-1.5/cblas/sswap.c
268	CB268	428	0	7	0.00	gsl-1.5/cblas/ssymm.c
269	CB269	387	0	7	0.00	gsl-1.5/cblas/ssymv.c
270	CB270	320	0	6	0.00	gsl-1.5/cblas/ssyr.c
271	CB271	356	0	6	0.00	gsl-1.5/cblas/ssyr2.c
272	CB272	453	0	8	0.00	gsl-1.5/cblas/ssyr2k.c
273	CB273	404	0	7	0.00	gsl-1.5/cblas/ssyrk.c
274	CB274	396	0	7	0.00	gsl-1.5/cblas/stbmv.c
275	CB275	396	0	7	0.00	gsl-1.5/cblas/stbsv.c
276	CB276	356	0	6	0.00	gsl-1.5/cblas/stpmv.c
277	CB277	356	0	6	0.00	gsl-1.5/cblas/stpsv.c
278	CB278	455	0	8	0.00	gsl-1.5/cblas/strmm.c
279	CB279	383	0	7	0.00	gsl-1.5/cblas/strmv.c
280	CB280	455	0	8	0.00	gsl-1.5/cblas/strsm.c
281	CB281	383	0	7	0.00	gsl-1.5/cblas/strsv.c
282	CB282	202	0	4	0.00	gsl-1.5/cblas/dasum.c
283	CB283	260	0	5	0.00	gsl-1.5/cblas/daxpy.c
284	CB284	240	0	5	0.00	gsl-1.5/cblas/dcopy.c
285	CB285	324	0	5	0.00	gsl-1.5/cblas/ddot.c
286	CB286	456	0	8	0.00	gsl-1.5/cblas/dgbmv.c
287	CB287	474	0	8	0.00	gsl-1.5/cblas/dgemm.c
288	CB288	415	0	7	0.00	gsl-1.5/cblas/dgemv.c
289	CB289	342	0	6	0.00	gsl-1.5/cblas/dger.c
290	CB290	202	0	4	0.00	gsl-1.5/cblas/dnrm2.c
291	CB291	261	0	5	0.00	gsl-1.5/cblas/drot.c
292	CB292	196	0	4	0.00	gsl-1.5/cblas/drotg.c
293	CB293	249	0	5	0.00	gsl-1.5/cblas/drotm.c
294	CB294	218	0	4	0.00	gsl-1.5/cblas/drotmg.c
295	CB295	406	0	7	0.00	gsl-1.5/cblas/dsbmv.c
296	CB296	214	0	4	0.00	gsl-1.5/cblas/dscal.c
297	CB297	323	0	5	0.00	gsl-1.5/cblas/dsdot.c
298	CB298	379	0	7	0.00	gsl-1.5/cblas/dspmv.c
299	CB299	310	0	6	0.00	gsl-1.5/cblas/dspr.c
300	CB300	347	0	6	0.00	gsl-1.5/cblas/dspr2.c
301	CB301	234	0	5	0.00	gsl-1.5/cblas/dswap.c
302	CB302	447	0	8	0.00	gsl-1.5/cblas/dsymm.c
303	CB303	393	0	7	0.00	gsl-1.5/cblas/dsymv.c

304	CB304	324	0	6	0.00	gsl-1.5/cblas/dsyr.c
305	CB305	361	0	6	0.00	gsl-1.5/cblas/dsyr2.c
306	CB306	459	0	8	0.00	gsl-1.5/cblas/dsyr2k.c
307	CB307	408	0	7	0.00	gsl-1.5/cblas/dsyrk.c
308	CB308	399	0	7	0.00	gsl-1.5/cblas/dtbmv.c
309	CB309	399	0	7	0.00	gsl-1.5/cblas/dtbsv.c
310	CB310	359	0	6	0.00	gsl-1.5/cblas/dtpmv.c
311	CB311	359	0	6	0.00	gsl-1.5/cblas/dtpsv.c
312	CB312	459	0	8	0.00	gsl-1.5/cblas/dtrmm.c
313	CB313	386	0	7	0.00	gsl-1.5/cblas/dtrmv.c
314	CB314	459	0	8	0.00	gsl-1.5/cblas/dtrsm.c
315	CB315	386	0	7	0.00	gsl-1.5/cblas/dtrsv.c
316	CB316	201	0	4	0.00	gsl-1.5/cblas/dzasum.c
317	CB317	201	0	4	0.00	gsl-1.5/cblas/dznrm2.c
318	CB318	254	0	5	0.00	gsl-1.5/cblas/caxpy.c
319	CB319	235	0	5	0.00	gsl-1.5/cblas/ccopy.c
320	CB320	300	0	5	0.00	gsl-1.5/cblas/cdotc_sub.c
321	CB321	297	0	5	0.00	gsl-1.5/cblas/cdotu_sub.c
322	CB322	434	0	7	0.00	gsl-1.5/cblas/cgbmv.c
323	CB323	465	0	8	0.00	gsl-1.5/cblas/cgemm.c
324	CB324	406	0	7	0.00	gsl-1.5/cblas/cgemv.c
325	CB325	338	0	6	0.00	gsl-1.5/cblas/cgerc.c
326	CB326	338	0	6	0.00	gsl-1.5/cblas/cgeru.c
327	CB327	397	0	7	0.00	gsl-1.5/cblas/chbmv.c
328	CB328	423	0	7	0.00	gsl-1.5/cblas/chemm.c
329	CB329	384	0	7	0.00	gsl-1.5/cblas/chemv.c
330	CB330	318	0	6	0.00	gsl-1.5/cblas/cher.c
331	CB331	353	0	6	0.00	gsl-1.5/cblas/cher2.c
332	CB332	434	0	7	0.00	gsl-1.5/cblas/cher2k.c
333	CB333	399	0	7	0.00	gsl-1.5/cblas/cherk.c
334	CB334	357	0	6	0.00	gsl-1.5/cblas/chpmv.c
335	CB335	304	0	6	0.00	gsl-1.5/cblas/chpr.c
336	CB336	339	0	6	0.00	gsl-1.5/cblas/chpr2.c
337	CB337	210	0	4	0.00	gsl-1.5/cblas/cscal.c
338	CB338	213	0	4	0.00	gsl-1.5/cblas/csscal.c
339	CB339	216	0	4	0.00	gsl-1.5/cblas/cswap.c
340	CB340	425	0	7	0.00	gsl-1.5/cblas/csymm.c
341	CB341	436	0	7	0.00	gsl-1.5/cblas/csyr2k.c
342	CB342	401	0	7	0.00	gsl-1.5/cblas/csyrk.c
343	CB343	394	0	7	0.00	gsl-1.5/cblas/ctbmv.c
344	CB344	414	0	7	0.00	gsl-1.5/cblas/ctbsv.c
345	CB345	354	0	6	0.00	gsl-1.5/cblas/ctpmv.c
346	CB346	374	0	6	0.00	gsl-1.5/cblas/ctpsv.c
347	CB347	453	0	8	0.00	gsl-1.5/cblas/ctrmm.c
348	CB348	381	0	7	0.00	gsl-1.5/cblas/ctrmv.c
349	CB349	473	0	8	0.00	gsl-1.5/cblas/ctrsm.c
350	CB350	401	0	7	0.00	gsl-1.5/cblas/ctrsv.c
351	CB351	255	0	5	0.00	gsl-1.5/cblas/zaxpy.c
352	CB352	236	0	5	0.00	gsl-1.5/cblas/zcopy.c
353	CB353	301	0	5	0.00	gsl-1.5/cblas/zdotc_sub.c
354	CB354	298	0	5	0.00	gsl-1.5/cblas/zdotu_sub.c
355	CB355	215	0	4	0.00	gsl-1.5/cblas/zdscal.c
356	CB356	435	0	7	0.00	gsl-1.5/cblas/zgbmv.c
357	CB357	466	0	8	0.00	gsl-1.5/cblas/zgemm.c
358	CB358	407	0	7	0.00	gsl-1.5/cblas/zgemv.c
359	CB359	339	0	6	0.00	gsl-1.5/cblas/zgerc.c
360	CB360	339	0	6	0.00	gsl-1.5/cblas/zgeru.c

361	CB361	398	0	7	0.00	gsl-1.5/cblas/zhbmv.c
362	CB362	424	0	7	0.00	gsl-1.5/cblas/zhemm.c
363	CB363	385	0	7	0.00	gsl-1.5/cblas/zhemv.c
364	CB364	320	0	6	0.00	gsl-1.5/cblas/zher.c
365	CB365	354	0	6	0.00	gsl-1.5/cblas/zher2.c
366	CB366	436	0	7	0.00	gsl-1.5/cblas/zher2k.c
367	CB367	402	0	7	0.00	gsl-1.5/cblas/zherk.c
368	CB368	358	0	6	0.00	gsl-1.5/cblas/zhpmv.c
369	CB369	306	0	6	0.00	gsl-1.5/cblas/zhpr.c
370	CB370	340	0	6	0.00	gsl-1.5/cblas/zhpr2.c
371	CB371	211	0	4	0.00	gsl-1.5/cblas/zscal.c
372	CB372	217	0	4	0.00	gsl-1.5/cblas/zswap.c
373	CB373	426	0	7	0.00	gsl-1.5/cblas/zsymm.c
374	CB374	437	0	7	0.00	gsl-1.5/cblas/zsyr2k.c
375	CB375	402	0	7	0.00	gsl-1.5/cblas/zsyrk.c
376	CB376	395	0	7	0.00	gsl-1.5/cblas/ztbmv.c
377	CB377	415	0	7	0.00	gsl-1.5/cblas/ztbsv.c
378	CB378	355	0	6	0.00	gsl-1.5/cblas/ztpmv.c
379	CB379	375	0	6	0.00	gsl-1.5/cblas/ztpsv.c
380	CB380	454	0	8	0.00	gsl-1.5/cblas/ztrmm.c
381	CB381	382	0	7	0.00	gsl-1.5/cblas/ztrmv.c
382	CB382	474	0	8	0.00	gsl-1.5/cblas/ztrsm.c
383	CB383	402	0	7	0.00	gsl-1.5/cblas/ztrsv.c
384	CB384	206	0	4	0.00	gsl-1.5/cblas/icamax.c
385	CB385	209	0	4	0.00	gsl-1.5/cblas/idamax.c
386	CB386	207	0	4	0.00	gsl-1.5/cblas/isamax.c
387	CB387	207	0	4	0.00	gsl-1.5/cblas/izamax.c
388	CB388	1179	18	13	1.38	gsl-1.5/cblas/xerbla.c
389	CB389	1030	18	6	3.00	gsl-1.5/cblas/test.c
390	CB390	2481	0	111	0.00	gsl-1.5/cblas/test_amax.c
391	CB391	2717	0	112	0.00	gsl-1.5/cblas/test_asum.c
392	CB392	5057	0	202	0.00	gsl-1.5/cblas/test_axpy.c
393	CB393	4634	0	190	0.00	gsl-1.5/cblas/test_copy.c
394	CB394	8786	0	322	0.00	gsl-1.5/cblas/test_dot.c
395	CB395	19476	0	496	0.00	gsl-1.5/cblas/test_gbmv.c
396	CB396	42319	0	1288	0.00	gsl-1.5/cblas/test_gemm.c
397	CB397	12647	0	456	0.00	gsl-1.5/cblas/test_gemv.c
398	CB398	6644	0	252	0.00	gsl-1.5/cblas/test_ger.c
399	CB399	14289	0	372	0.00	gsl-1.5/cblas/test_hbmv.c
400	CB400	12084	0	388	0.00	gsl-1.5/cblas/test_hemm.c
401	CB401	10217	0	356	0.00	gsl-1.5/cblas/test_hemv.c
402	CB402	4308	0	156	0.00	gsl-1.5/cblas/test_her.c
403	CB403	4925	0	172	0.00	gsl-1.5/cblas/test_her2.c
404	CB404	11205	0	388	0.00	gsl-1.5/cblas/test_her2k.c
405	CB405	11400	0	356	0.00	gsl-1.5/cblas/test_herk.c
406	CB406	11369	0	340	0.00	gsl-1.5/cblas/test_hpmv.c
407	CB407	4874	0	148	0.00	gsl-1.5/cblas/test_hpr.c
408	CB408	4809	0	164	0.00	gsl-1.5/cblas/test_hpr2.c
409	CB409	2746	0	112	0.00	gsl-1.5/cblas/test_nrm2.c
410	CB410	12922	0	580	0.00	gsl-1.5/cblas/test_rot.c
411	CB411	47807	0	1474	0.00	gsl-1.5/cblas/test_rotg.c
412	CB412	35349	0	1384	0.00	gsl-1.5/cblas/test_rotm.c
413	CB413	6337	0	148	0.00	gsl-1.5/cblas/test_rotmg.c
414	CB414	9917	0	356	0.00	gsl-1.5/cblas/test_sbmv.c
415	CB415	20978	0	796	0.00	gsl-1.5/cblas/test_scal.c
416	CB416	8329	0	324	0.00	gsl-1.5/cblas/test_spmv.c
417	CB417	3652	0	140	0.00	gsl-1.5/cblas/test_spr.c

418	CB418	4105	0	156	0.00	gsl-1.5/cblas/test_spr2.c
419	CB419	7247	0	280	0.00	gsl-1.5/cblas/test_swap.c
420	CB420	20972	0	756	0.00	gsl-1.5/cblas/test_symm.c
421	CB421	7937	0	340	0.00	gsl-1.5/cblas/test_symv.c
422	CB422	3416	0	148	0.00	gsl-1.5/cblas/test_syr.c
423	CB423	3817	0	164	0.00	gsl-1.5/cblas/test_syr2.c
424	CB424	20076	0	756	0.00	gsl-1.5/cblas/test_syr2k.c
425	CB425	20398	0	692	0.00	gsl-1.5/cblas/test_syrk.c
426	CB426	53793	0	1652	0.00	gsl-1.5/cblas/test_tbmv.c
427	CB427	54428	0	1652	0.00	gsl-1.5/cblas/test_tbsv.c
428	CB428	42537	0	1492	0.00	gsl-1.5/cblas/test_tpmv.c
429	CB429	42777	0	1492	0.00	gsl-1.5/cblas/test_tpsv.c
430	CB430	125454	0	3620	0.00	gsl-1.5/cblas/test_trmm.c
431	CB431	39593	0	1572	0.00	gsl-1.5/cblas/test_trmv.c
432	CB432	125775	0	3620	0.00	gsl-1.5/cblas/test_trsm.c
433	CB433	39777	0	1572	0.00	gsl-1.5/cblas/test_trsv.c
434	BL434	21913	50	427	0.12	gsl-1.5/blas/gsl_blas.h
435	BL435	1558	25	8	3.13	gsl-1.5/blas/gsl_blas_types.h
436	BL436	56926	74	1788	0.04	gsl-1.5/blas/blas.c
437	LI437	1358	21	23	0.91	gsl-1.5/linalg/givens.c
438	LI438	1904	22	28	0.79	gsl-1.5/linalg/apply_givens.c
439	LI439	9918	30	325	0.09	gsl-1.5/linalg/svdstep.c
440	LI440	1930	25	32	0.78	gsl-1.5/linalg/tridiag.h
441	LI441	16202	100	249	0.40	gsl-1.5/linalg/gsl_linalg.h
442	LI442	4073	19	98	0.19	gsl-1.5/linalg/multiply.c
443	LI443	4997	51	101	0.50	gsl-1.5/linalg/exponential.c
444	LI444	15129	82	418	0.20	gsl-1.5/linalg/tridiag.c
445	LI445	7402	49	197	0.25	gsl-1.5/linalg/lu.c
446	LI446	8698	49	213	0.23	gsl-1.5/linalg/luc.c
447	LI447	4630	28	114	0.25	gsl-1.5/linalg/hh.c
448	LI448	13381	107	327	0.33	gsl-1.5/linalg/qr.c
449	LI449	12822	94	299	0.31	gsl-1.5/linalg/qrpt.c
450	LI450	17242	114	390	0.29	gsl-1.5/linalg/svd.c
451	LI451	7573	43	141	0.30	gsl-1.5/linalg/householder.c
452	LI452	5947	46	115	0.40	gsl-1.5/linalg/householdercomplex.c
453	LI453	5418	40	124	0.32	gsl-1.5/linalg/cholesky.c
454	LI454	6657	61	130	0.47	gsl-1.5/linalg/symmtd.c
455	LI455	7298	57	138	0.41	gsl-1.5/linalg/hermtd.c
456	LI456	10115	82	215	0.38	gsl-1.5/linalg/bidiag.c
457	LI457	1933	24	38	0.63	gsl-1.5/linalg/balance.c
458	LI458	70191	51	1849	0.03	gsl-1.5/linalg/test.c
459	EI459	3202	8	130	0.06	gsl-1.5/eigen/qrstep.c
460	EI460	3733	36	64	0.56	gsl-1.5/eigen/gsl_eigen.h
461	EI461	7473	38	187	0.20	gsl-1.5/eigen/jacobi.c
462	EI462	4683	36	99	0.36	gsl-1.5/eigen/symm.c
463	EI463	5880	36	129	0.28	gsl-1.5/eigen/symmv.c
464	EI464	4719	32	103	0.31	gsl-1.5/eigen/herm.c
465	EI465	7065	36	157	0.23	gsl-1.5/eigen/hermv.c
466	EI466	4624	27	114	0.24	gsl-1.5/eigen/sort.c
467	EI467	11954	32	302	0.11	gsl-1.5/eigen/test.c
468	SP468	1506	27	6	4.50	gsl-1.5/specfunc/bessel_amp_phase.h
469	SP469	1161	20	3	6.67	gsl-1.5/specfunc/bessel_olver.h
470	SP470	1236	20	7	2.86	gsl-1.5/specfunc/bessel_temme.h
471	SP471	3087	38	29	1.31	gsl-1.5/specfunc/bessel.h
472	SP472	2249	42	16	2.63	gsl-1.5/specfunc/hyperg.h
473	SP473	2346	46	14	3.29	gsl-1.5/specfunc/legendre.h
474	SP474	416	1	0	0.00	gsl-1.5/specfunc/eval.h

475	SP475	1189	24	8	3.00	gsl-1.5/specfunc/chebyshev.h
476	SP476	656	0	26	0.00	gsl-1.5/specfunc/cheb_eval.c
477	SP477	680	0	25	0.00	gsl-1.5/specfunc/cheb_eval_mode.c
478	SP478	126	1	0	0.00	gsl-1.5/specfunc/check.h
479	SP479	1821	0	0	0.00	gsl-1.5/specfunc/error.h
480	SP480	1008	2	0	0.00	gsl-1.5/specfunc/gsl_sf.h
481	SP481	3688	68	26	2.62	gsl-1.5/specfunc/gsl_sf_airy.h
482	SP482	14047	311	102	3.05	gsl-1.5/specfunc/gsl_sf_bessel.h
483	SP483	1446	26	4	6.50	gsl-1.5/specfunc/gsl_sf_clausen.h
484	SP484	4381	55	39	1.41	gsl-1.5/specfunc/gsl_sf_coulomb.h
485	SP485	4144	56	39	1.44	gsl-1.5/specfunc/gsl_sf_coupling.h
486	SP486	1381	26	4	6.50	gsl-1.5/specfunc/gsl_sf_dawson.h
487	SP487	1882	37	10	3.70	gsl-1.5/specfunc/gsl_sf_debye.h
488	SP488	1687	29	5	5.80	gsl-1.5/specfunc/gsl_sf_dilog.h
489	SP489	1636	28	5	5.60	gsl-1.5/specfunc/gsl_sf_elementary.h
490	SP490	3870	50	22	2.27	gsl-1.5/specfunc/gsl_sf_ellint.h
491	SP491	1343	25	3	8.33	gsl-1.5/specfunc/gsl_sf_elljac.h
492	SP492	2274	47	14	3.36	gsl-1.5/specfunc/gsl_sf_erf.h
493	SP493	4335	71	20	3.55	gsl-1.5/specfunc/gsl_sf_exp.h
494	SP494	3819	79	26	3.04	gsl-1.5/specfunc/gsl_sf_expint.h
495	SP495	3392	66	20	3.30	gsl-1.5/specfunc/gsl_sf_fermi_dirac.h
496	SP496	7416	172	45	3.82	gsl-1.5/specfunc/gsl_sf_gamma.h
497	SP497	2133	36	11	3.27	gsl-1.5/specfunc/gsl_sf_gegenbauer.h
498	SP498	4496	86	24	3.58	gsl-1.5/specfunc/gsl_sf_hyperg.h
499	SP499	1979	32	10	3.20	gsl-1.5/specfunc/gsl_sf_laguerre.h
500	SP500	1792	39	6	6.50	gsl-1.5/specfunc/gsl_sf_lambert.h
501	SP501	8700	180	67	2.69	gsl-1.5/specfunc/gsl_sf_legendre.h
502	SP502	2846	43	11	3.91	gsl-1.5/specfunc/gsl_sf_log.h
503	SP503	1369	23	4	5.75	gsl-1.5/specfunc/gsl_sf_pow_int.h
504	SP504	2437	53	14	3.79	gsl-1.5/specfunc/gsl_sf_psi.h
505	SP505	1556	20	14	1.43	gsl-1.5/specfunc/gsl_sf_result.h
506	SP506	1702	30	6	5.00	gsl-1.5/specfunc/gsl_sf_synchrotron.h
507	SP507	1957	39	10	3.90	gsl-1.5/specfunc/gsl_sf_transport.h
508	SP508	3936	74	27	2.74	gsl-1.5/specfunc/gsl_sf_trig.h
509	SP509	2873	62	16	3.88	gsl-1.5/specfunc/gsl_sf_zeta.h
510	SP510	173	3	0	0.00	gsl-1.5/specfunc/gsl_specfunc.h
511	SP511	23525	157	640	0.25	gsl-1.5/specfunc/airy.c
512	SP512	25608	143	682	0.21	gsl-1.5/specfunc/airy_der.c
513	SP513	13616	25	482	0.05	gsl-1.5/specfunc/airy_zero.c
514	SP514	2987	22	72	0.31	gsl-1.5/specfunc/atanint.c
515	SP515	26921	143	536	0.27	gsl-1.5/specfunc/bessel.c
516	SP516	6239	46	154	0.30	gsl-1.5/specfunc/bessel_I0.c
517	SP517	6791	45	179	0.25	gsl-1.5/specfunc/bessel_I1.c
518	SP518	6391	32	161	0.20	gsl-1.5/specfunc/bessel_In.c
519	SP519	3395	27	64	0.42	gsl-1.5/specfunc/bessel_Inu.c
520	SP520	3241	33	52	0.63	gsl-1.5/specfunc/bessel_J0.c
521	SP521	3658	37	63	0.59	gsl-1.5/specfunc/bessel_J1.c
522	SP522	5045	29	137	0.21	gsl-1.5/specfunc/bessel_Jn.c
523	SP523	5189	39	81	0.48	gsl-1.5/specfunc/bessel_Jnu.c
524	SP524	5933	45	137	0.33	gsl-1.5/specfunc/bessel_K0.c
525	SP525	6100	45	144	0.31	gsl-1.5/specfunc/bessel_K1.c
526	SP526	6569	36	162	0.22	gsl-1.5/specfunc/bessel_Kn.c
527	SP527	4662	43	88	0.49	gsl-1.5/specfunc/bessel_Knu.c
528	SP528	3597	35	60	0.58	gsl-1.5/specfunc/bessel_Y0.c
529	SP529	4156	33	78	0.42	gsl-1.5/specfunc/bessel_Y1.c
530	SP530	5479	26	148	0.18	gsl-1.5/specfunc/bessel_Yn.c
531	SP531	3376	38	42	0.90	gsl-1.5/specfunc/bessel_Ynu.c

532	SP532	4630	26	145	0.18	gsl-1.5/specfunc/bessel_amp_phase.c
533	SP533	8481	28	232	0.12	gsl-1.5/specfunc/bessel_i.c
534	SP534	11015	36	312	0.12	gsl-1.5/specfunc/bessel_j.c
535	SP535	6266	31	170	0.18	gsl-1.5/specfunc/bessel_k.c
536	SP536	31807	59	857	0.07	gsl-1.5/specfunc/bessel_olver.c
537	SP537	6292	26	162	0.16	gsl-1.5/specfunc/bessel_temme.c
538	SP538	7644	33	206	0.16	gsl-1.5/specfunc/bessel_y.c
539	SP539	28785	93	1016	0.09	gsl-1.5/specfunc/bessel_zero.c
540	SP540	4254	42	82	0.51	gsl-1.5/specfunc/bessel_sequence.c
541	SP541	3897	24	84	0.29	gsl-1.5/specfunc/beta.c
542	SP542	5076	28	118	0.24	gsl-1.5/specfunc/beta_inc.c
543	SP543	2717	25	61	0.41	gsl-1.5/specfunc/clausen.c
544	SP544	39349	211	901	0.23	gsl-1.5/specfunc/coulomb.c
545	SP545	13585	25	336	0.07	gsl-1.5/specfunc/coupling.c
546	SP546	3675	22	72	0.31	gsl-1.5/specfunc/coulomb_bound.c
547	SP547	9991	45	214	0.21	gsl-1.5/specfunc/dawson.c
548	SP548	9440	25	321	0.08	gsl-1.5/specfunc/debye.c
549	SP549	14757	100	357	0.28	gsl-1.5/specfunc/dilog.c
550	SP550	2440	24	43	0.56	gsl-1.5/specfunc/elementary.c
551	SP551	15407	58	388	0.15	gsl-1.5/specfunc/ellint.c
552	SP552	2913	24	68	0.35	gsl-1.5/specfunc/elljac.c
553	SP553	12207	58	314	0.18	gsl-1.5/specfunc/erfc.c
554	SP554	15838	46	495	0.09	gsl-1.5/specfunc/exp.c
555	SP555	13804	73	374	0.20	gsl-1.5/specfunc/expint.c
556	SP556	3448	22	101	0.22	gsl-1.5/specfunc/expint3.c
557	SP557	34389	150	1322	0.11	gsl-1.5/specfunc/fermi_dirac.c
558	SP558	4971	30	129	0.23	gsl-1.5/specfunc/gegenbauer.c
559	SP559	60609	179	1353	0.13	gsl-1.5/specfunc/gamma.c
560	SP560	18643	136	449	0.30	gsl-1.5/specfunc/gamma_inc.c
561	SP561	5100	32	118	0.27	gsl-1.5/specfunc/hyperg_0F1.c
562	SP562	1862	25	24	1.04	gsl-1.5/specfunc/hyperg_2F0.c
563	SP563	59055	374	1266	0.30	gsl-1.5/specfunc/hyperg_1F1.c
564	SP564	27638	115	700	0.16	gsl-1.5/specfunc/hyperg_2F1.c
565	SP565	44780	189	1083	0.17	gsl-1.5/specfunc/hyperg_U.c
566	SP566	8581	33	210	0.16	gsl-1.5/specfunc/hyperg.c
567	SP567	9390	55	224	0.25	gsl-1.5/specfunc/laguerre.c
568	SP568	6116	57	141	0.40	gsl-1.5/specfunc/lambert.c
569	SP569	17546	81	378	0.21	gsl-1.5/specfunc/legendre_H3d.c
570	SP570	9696	43	273	0.16	gsl-1.5/specfunc/legendre_Qn.c
571	SP571	43218	140	1047	0.13	gsl-1.5/specfunc/legendre_con.c
572	SP572	20268	91	574	0.16	gsl-1.5/specfunc/legendre_poly.c
573	SP573	6901	42	186	0.23	gsl-1.5/specfunc/log.c
574	SP574	13160	90	298	0.30	gsl-1.5/specfunc/poch.c
575	SP575	1893	26	28	0.93	gsl-1.5/specfunc/pow_int.c
576	SP576	21029	92	577	0.16	gsl-1.5/specfunc/psi.c
577	SP577	7825	32	0	0.00	gsl-1.5/specfunc/recurse.h
578	SP578	2380	53	25	2.12	gsl-1.5/specfunc/result.c
579	SP579	3955	31	81	0.38	gsl-1.5/specfunc/shint.c
580	SP580	11122	81	279	0.29	gsl-1.5/specfunc/sinint.c
581	SP581	6765	25	220	0.11	gsl-1.5/specfunc/synchrotron.c
582	SP582	12781	25	423	0.06	gsl-1.5/specfunc/transport.c
583	SP583	19180	76	529	0.14	gsl-1.5/specfunc/trig.c
584	SP584	33032	103	770	0.13	gsl-1.5/specfunc/zeta.c
585	SP585	98139	85	1396	0.06	gsl-1.5/specfunc/test_sf.c
586	SP586	3907	23	17	1.35	gsl-1.5/specfunc/test_sf.h
587	SP587	9327	27	80	0.34	gsl-1.5/specfunc/test_airy.c
588	SP588	38562	23	444	0.05	gsl-1.5/specfunc/test_bessel.c

589	SP589	15174	25	313	0.08	gsl-1.5/specfunc/test_coulomb.c
590	SP590	5225	25	75	0.33	gsl-1.5/specfunc/test_dilog.c
591	SP591	22942	25	236	0.11	gsl-1.5/specfunc/test_gamma.c
592	SP592	40354	77	345	0.22	gsl-1.5/specfunc/test_hyperg.c
593	SP593	34943	27	415	0.07	gsl-1.5/specfunc/test_legendre.c
594	RN594	1853	25	24	1.04	gsl-1.5/rng/schrage.c
595	RN595	6745	22	101	0.22	gsl-1.5/rng/gsl_rng.h
596	RN596	2180	36	40	0.90	gsl-1.5/rng/borosh13.c
597	RN597	5293	78	93	0.84	gsl-1.5/rng/cmrg.c
598	RN598	2225	34	42	0.81	gsl-1.5/rng/coveyou.c
599	RN599	2366	21	47	0.45	gsl-1.5/rng/default.c
600	RN600	1384	18	24	0.75	gsl-1.5/rng/file.c
601	RN601	2312	37	40	0.93	gsl-1.5/rng/fishman18.c
602	RN602	2333	35	51	0.69	gsl-1.5/rng/fishman20.c
603	RN603	2872	39	57	0.68	gsl-1.5/rng/fishman2x.c
604	RN604	5171	69	65	1.06	gsl-1.5/rng/gfsr4.c
605	RN605	2533	36	47	0.77	gsl-1.5/rng/knuthran2.c
606	RN606	4812	54	106	0.51	gsl-1.5/rng/knuthran.c
607	RN607	2254	33	45	0.73	gsl-1.5/rng/lecuyer21.c
608	RN608	2844	44	49	0.90	gsl-1.5/rng/minstd.c
609	RN609	3941	54	71	0.76	gsl-1.5/rng/mrg.c
610	RN610	6558	81	122	0.66	gsl-1.5/rng/mt.c
611	RN611	4601	72	77	0.94	gsl-1.5/rng/r250.c
612	RN612	2573	29	53	0.55	gsl-1.5/rng/ran0.c
613	RN613	2983	26	79	0.33	gsl-1.5/rng/ran1.c
614	RN614	3545	32	86	0.37	gsl-1.5/rng/ran2.c
615	RN615	2990	27	76	0.36	gsl-1.5/rng/ran3.c
616	RN616	3811	38	77	0.49	gsl-1.5/rng/rand48.c
617	RN617	2267	35	36	0.97	gsl-1.5/rng/rand.c
618	RN618	16384	89	492	0.18	gsl-1.5/rng/random.c
619	RN619	2551	41	39	1.05	gsl-1.5/rng/randu.c
620	RN620	4197	44	89	0.49	gsl-1.5/rng/ranf.c
621	RN621	5369	59	136	0.43	gsl-1.5/rng/ranlux.c
622	RN622	5745	34	174	0.20	gsl-1.5/rng/ranlxd.c
623	RN623	7086	41	217	0.19	gsl-1.5/rng/ranlxs.c
624	RN624	3933	45	96	0.47	gsl-1.5/rng/ranmar.c
625	RN625	4003	22	137	0.16	gsl-1.5/rng/rng.c
626	RN626	7540	141	49	2.88	gsl-1.5/rng/slatec.c
627	RN627	5529	86	77	1.12	gsl-1.5/rng/taus.c
628	RN628	5468	77	68	1.13	gsl-1.5/rng/taus113.c
629	RN629	2257	32	38	0.84	gsl-1.5/rng/transputer.c
630	RN630	3826	39	82	0.48	gsl-1.5/rng/tt.c
631	RN631	2636	18	69	0.26	gsl-1.5/rng/types.c
632	RN632	6168	109	77	1.42	gsl-1.5/rng/uni32.c
633	RN633	6047	109	76	1.43	gsl-1.5/rng/uni.c
634	RN634	2159	33	36	0.92	gsl-1.5/rng/vax.c
635	RN635	2172	36	40	0.90	gsl-1.5/rng/waterman14.c
636	RN636	3502	38	81	0.47	gsl-1.5/rng/zuf.c
637	RN637	15668	72	386	0.19	gsl-1.5/rng/test.c
638	RA638	8267	20	102	0.20	gsl-1.5/randist/gsl_randist.h
639	RA639	1361	23	29	0.79	gsl-1.5/randist/bernoulli.c
640	RA640	1651	23	24	0.96	gsl-1.5/randist/beta.c
641	RA641	2112	26	29	0.90	gsl-1.5/randist/bigauss.c
642	RA642	2008	24	47	0.51	gsl-1.5/randist/binomial.c
643	RA643	1395	23	18	1.28	gsl-1.5/randist/cauchy.c
644	RA644	1465	23	21	1.10	gsl-1.5/randist/chisq.c
645	RA645	2739	35	47	0.74	gsl-1.5/randist/dirichlet.c

646	RA646	13148	173	172	1.01	gsl-1.5/randist/discrete.c
647	RA647	1484	24	20	1.20	gsl-1.5/randist/erlang.c
648	RA648	1326	23	19	1.21	gsl-1.5/randist/exponential.c
649	RA649	3582	47	65	0.72	gsl-1.5/randist/exppow.c
650	RA650	1914	25	27	0.93	gsl-1.5/randist/fdist.c
651	RA651	1389	25	18	1.39	gsl-1.5/randist/flat.c
652	RA652	3976	39	104	0.38	gsl-1.5/randist/gamma.c
653	RA653	3058	36	52	0.69	gsl-1.5/randist/gauss.c
654	RA654	2632	28	58	0.48	gsl-1.5/randist/gausstail.c
655	RA655	1597	24	32	0.75	gsl-1.5/randist/geometric.c
656	RA656	1813	27	33	0.82	gsl-1.5/randist/gumbel.c
657	RA657	2810	30	74	0.41	gsl-1.5/randist/hyperg.c
658	RA658	1452	23	24	0.96	gsl-1.5/randist/laplace.c
659	RA659	3716	61	55	1.11	gsl-1.5/randist/levy.c
660	RA660	1844	24	41	0.59	gsl-1.5/randist/logarithmic.c
661	RA661	1377	23	19	1.21	gsl-1.5/randist/logistic.c
662	RA662	1904	26	29	0.90	gsl-1.5/randist/lognormal.c
663	RA663	3009	39	59	0.66	gsl-1.5/randist/multinomial.c
664	RA664	1664	25	17	1.47	gsl-1.5/randist/nbinomial.c
665	RA665	1356	23	20	1.15	gsl-1.5/randist/pareto.c
666	RA666	1658	29	12	2.42	gsl-1.5/randist/pascal.c
667	RA667	2054	24	48	0.50	gsl-1.5/randist/poisson.c
668	RA668	2019	28	41	0.68	gsl-1.5/randist/rayleigh.c
669	RA669	3512	32	69	0.46	gsl-1.5/randist/shuffle.c
670	RA670	3460	48	58	0.83	gsl-1.5/randist/sphere.c
671	RA671	2149	27	34	0.79	gsl-1.5/randist/tdist.c
672	RA672	1551	23	31	0.74	gsl-1.5/randist/weibull.c
673	RA673	21051	34	369	0.09	gsl-1.5/randist/landau.c
674	RA674	12738	159	145	1.10	gsl-1.5/randist/binomial_tpe.c
675	RA675	35624	32	1467	0.02	gsl-1.5/randist/test.c
676	FF676	4830	18	79	0.23	gsl-1.5/fft/c_pass.h
677	FF677	3290	18	47	0.38	gsl-1.5/fft/hc_pass.h
678	FF678	3408	18	42	0.43	gsl-1.5/fft/real_pass.h
679	FF679	2957	18	38	0.47	gsl-1.5/fft/signals.h
680	FF680	7236	29	200	0.14	gsl-1.5/fft/signals_source.c
681	FF681	7078	21	178	0.12	gsl-1.5/fft/c_main.c
682	FF682	4716	26	126	0.21	gsl-1.5/fft/c_init.c
683	FF683	3035	26	59	0.44	gsl-1.5/fft/c_pass_2.c
684	FF684	4388	31	78	0.40	gsl-1.5/fft/c_pass_3.c
685	FF685	5226	34	93	0.37	gsl-1.5/fft/c_pass_4.c
686	FF686	7674	43	129	0.33	gsl-1.5/fft/c_pass_5.c
687	FF687	8310	48	137	0.35	gsl-1.5/fft/c_pass_6.c
688	FF688	12744	66	190	0.35	gsl-1.5/fft/c_pass_7.c
689	FF689	6069	20	155	0.13	gsl-1.5/fft/c_pass_n.c
690	FF690	8414	33	212	0.16	gsl-1.5/fft/c_radix2.c
691	FF691	2498	22	57	0.39	gsl-1.5/fft/bitreverse.c
692	FF692	1324	18	8	2.25	gsl-1.5/fft/bitreverse.h
693	FF693	3700	25	111	0.23	gsl-1.5/fft/factorize.c
694	FF694	1206	18	5	3.60	gsl-1.5/fft/factorize.h
695	FF695	3272	24	77	0.31	gsl-1.5/fft/hc_init.c
696	FF696	3294	21	65	0.32	gsl-1.5/fft/hc_pass_2.c
697	FF697	5332	25	102	0.25	gsl-1.5/fft/hc_pass_3.c
698	FF698	6551	28	123	0.23	gsl-1.5/fft/hc_pass_4.c
699	FF699	9682	41	171	0.24	gsl-1.5/fft/hc_pass_5.c
700	FF700	8218	19	196	0.10	gsl-1.5/fft/hc_pass_n.c
701	FF701	4802	29	109	0.27	gsl-1.5/fft/hc_radix2.c
702	FF702	2851	18	56	0.32	gsl-1.5/fft/hc_unpack.c

703	FF703	876	0	0	0.00	gsl-1.5/fft/real.c
704	FF704	4321	26	116	0.22	gsl-1.5/fft/real_init.c
705	FF705	3691	23	69	0.33	gsl-1.5/fft/real_pass_2.c
706	FF706	5816	29	104	0.28	gsl-1.5/fft/real_pass_3.c
707	FF707	7178	35	128	0.27	gsl-1.5/fft/real_pass_4.c
708	FF708	10463	49	176	0.28	gsl-1.5/fft/real_pass_5.c
709	FF709	8503	19	200	0.10	gsl-1.5/fft/real_pass_n.c
710	FF710	3908	28	78	0.36	gsl-1.5/fft/real_radix2.c
711	FF711	1297	18	17	1.06	gsl-1.5/fft/real_unpack.c
712	FF712	1327	18	10	1.80	gsl-1.5/fft/compare.h
713	FF713	3224	18	88	0.20	gsl-1.5/fft/compare_source.c
714	FF714	3137	21	65	0.32	gsl-1.5/fft/dft_source.c
715	FF715	5237	21	136	0.15	gsl-1.5/fft/hc_main.c
716	FF716	3983	20	101	0.20	gsl-1.5/fft/real_main.c
717	FF717	15210	35	321	0.11	gsl-1.5/fft/test_complex_source.c
718	FF718	7535	26	152	0.17	gsl-1.5/fft/test_real_source.c
719	FF719	5269	24	72	0.33	gsl-1.5/fft/test_trap_source.c
720	FF720	943	18	6	3.00	gsl-1.5/fft/urand.c
721	FF721	366	3	0	0.00	gsl-1.5/fft/complex_internal.h
722	FF722	1359	24	7	3.43	gsl-1.5/fft/gsl_fft.h
723	FF723	5028	21	73	0.29	gsl-1.5/fft/gsl_fft_complex.h
724	FF724	3001	19	34	0.56	gsl-1.5/fft/gsl_fft_halfcomplex.h
725	FF725	2205	19	28	0.68	gsl-1.5/fft/gsl_fft_real.h
726	FF726	1787	19	10	1.90	gsl-1.5/fft/gsl_dft_complex.h
727	FF727	1827	19	10	1.90	gsl-1.5/fft/gsl_dft_complex_float.h
728	FF728	5577	21	73	0.29	gsl-1.5/fft/gsl_fft_complex_float.h
729	FF729	3207	19	34	0.56	gsl-1.5/fft/gsl_fft_halfcomplex_float.h
730	FF730	2355	19	28	0.68	gsl-1.5/fft/gsl_fft_real_float.h
731	FF731	474	0	0	0.00	gsl-1.5/fft/dft.c
732	FF732	2472	0	0	0.00	gsl-1.5/fft/fft.c
733	FF733	3243	21	50	0.42	gsl-1.5/fft/test.c
734	FF734	507	0	0	0.00	gsl-1.5/fft/signals.c
735	PO735	3150	20	85	0.24	gsl-1.5/poly/balance.c
736	PO736	1177	18	13	1.38	gsl-1.5/poly/companion.c
737	PO737	1182	18	19	0.95	gsl-1.5/poly/norm.c
738	PO738	5441	34	158	0.22	gsl-1.5/poly/qr.c
739	PO739	3593	40	35	1.14	gsl-1.5/poly/gsl_poly.h
740	PO740	2290	20	57	0.35	gsl-1.5/poly/dd.c
741	PO741	1176	20	8	2.50	gsl-1.5/poly/eval.c
742	PO742	1794	19	43	0.44	gsl-1.5/poly/solve_quadratic.c
743	PO743	2981	26	64	0.41	gsl-1.5/poly/solve_cubic.c
744	PO744	2355	19	56	0.34	gsl-1.5/poly/zsolve_quadratic.c
745	PO745	4213	26	97	0.27	gsl-1.5/poly/zsolve_cubic.c
746	PO746	2127	22	33	0.67	gsl-1.5/poly/zsolve.c
747	PO747	1798	19	32	0.59	gsl-1.5/poly/zsolve_init.c
748	PO748	16151	25	323	0.08	gsl-1.5/poly/test.c
749	FI749	3243	20	51	0.39	gsl-1.5/fit/gsl_fit.h
750	FI750	8374	59	208	0.28	gsl-1.5/fit/linear.c
751	FI751	6184	12	118	0.10	gsl-1.5/fit/test.c
752	MU752	2755	1	110	0.01	gsl-1.5/multifit/lmutil.c
753	MU753	10371	40	222	0.18	gsl-1.5/multifit/lmpar.c
754	MU754	967	3	29	0.10	gsl-1.5/multifit/lmset.c
755	MU755	4971	17	134	0.13	gsl-1.5/multifit/lmiterate.c
756	MU756	5799	56	116	0.48	gsl-1.5/multifit/qrsolv.c
757	MU757	3077	0	98	0.00	gsl-1.5/multifit/test_brown.c
758	MU758	4837	0	256	0.00	gsl-1.5/multifit/test_enso.c
759	MU759	13966	1	170	0.01	gsl-1.5/multifit/test_filip.c

760	MU760	763	0	24	0.00	gsl-1.5/multifit/test_fn.c
761	MU761	8769	1	548	0.00	gsl-1.5/multifit/test_hahn1.c
762	MU762	6082	1	372	0.00	gsl-1.5/multifit/test_kirby2.c
763	MU763	7448	0	132	0.00	gsl-1.5/multifit/test_longley.c
764	MU764	6239	1	189	0.01	gsl-1.5/multifit/test_nelson.c
765	MU765	4726	0	100	0.00	gsl-1.5/multifit/test_pontius.c
766	MU766	2183	21	34	0.62	gsl-1.5/multifit/gsl_multifit.h
767	MU767	5829	27	96	0.28	gsl-1.5/multifit/gsl_multifit_nlin.h
768	MU768	8377	42	209	0.20	gsl-1.5/multifit/multilinear.c
769	MU769	3328	20	90	0.22	gsl-1.5/multifit/work.c
770	MU770	9599	20	278	0.07	gsl-1.5/multifit/lmder.c
771	MU771	3681	20	100	0.20	gsl-1.5/multifit/fsolver.c
772	MU772	4008	20	111	0.18	gsl-1.5/multifit/fdfsolver.c
773	MU773	2012	18	51	0.35	gsl-1.5/multifit/convergence.c
774	MU774	1122	18	7	2.57	gsl-1.5/multifit/gradient.c
775	MU775	4539	30	116	0.26	gsl-1.5/multifit/covar.c
776	MU776	5064	4	124	0.03	gsl-1.5/multifit/test.c
777	ST777	1192	20	11	1.82	gsl-1.5/statistics/mean_source.c
778	ST778	2821	20	52	0.38	gsl-1.5/statistics/variance_source.c
779	ST779	2949	20	45	0.44	gsl-1.5/statistics/covariance_source.c
780	ST780	1549	20	22	0.91	gsl-1.5/statistics/absdev_source.c
781	ST781	1746	22	21	1.05	gsl-1.5/statistics/skew_source.c
782	ST782	1929	23	24	0.96	gsl-1.5/statistics/kurtosis_source.c
783	ST783	1698	20	23	0.87	gsl-1.5/statistics/lag1_source.c
784	ST784	1433	20	12	1.67	gsl-1.5/statistics/p_variance_source.c
785	ST785	3460	26	92	0.28	gsl-1.5/statistics/minmax_source.c
786	ST786	1670	24	12	2.00	gsl-1.5/statistics/ttest_source.c
787	ST787	1329	18	20	0.90	gsl-1.5/statistics/median_source.c
788	ST788	1466	18	22	0.82	gsl-1.5/statistics/quantiles_source.c
789	ST789	1405	25	17	1.47	gsl-1.5/statistics/wmean_source.c
790	ST790	4046	22	80	0.28	gsl-1.5/statistics/wvariance_source.c
791	ST791	1694	20	23	0.87	gsl-1.5/statistics/wabsdev_source.c
792	ST792	2028	22	27	0.81	gsl-1.5/statistics/wskew_source.c
793	ST793	2205	23	30	0.77	gsl-1.5/statistics/wkurtosis_source.c
794	ST794	9585	19	218	0.09	gsl-1.5/statistics/test_float_source.c
795	ST795	7899	19	185	0.10	gsl-1.5/statistics/test_int_source.c
796	ST796	519	1	0	0.00	gsl-1.5/statistics/gsl_statistics.h
797	ST797	4181	19	29	0.66	gsl-1.5/statistics/gsl_statistics_char.h
798	ST798	6044	21	42	0.50	gsl-1.5/statistics/gsl_statistics_double.h
799	ST799	6219	21	42	0.50	gsl-1.5/statistics/gsl_statistics_float.h
800	ST800	4115	19	29	0.66	gsl-1.5/statistics/gsl_statistics_int.h
801	ST801	4181	19	29	0.66	gsl-1.5/statistics/gsl_statistics_long.h
802	ST802	6849	21	42	0.50	gsl-1.5/statistics/gsl_statistics_long_double.h
803	ST803	4247	19	29	0.66	gsl-1.5/statistics/gsl_statistics_short.h
804	ST804	4527	19	29	0.66	gsl-1.5/statistics/gsl_statistics_uchar.h
805	ST805	4461	19	29	0.66	gsl-1.5/statistics/gsl_statistics_uint.h
806	ST806	4527	19	29	0.66	gsl-1.5/statistics/gsl_statistics_ulong.h
807	ST807	4593	19	29	0.66	gsl-1.5/statistics/gsl_statistics_ushort.h
808	ST808	1347	0	0	0.00	gsl-1.5/statistics/mean.c
809	ST809	1410	0	0	0.00	gsl-1.5/statistics/variance.c
810	ST810	1386	0	0	0.00	gsl-1.5/statistics/absdev.c
811	ST811	1366	0	0	0.00	gsl-1.5/statistics/skew.c
812	ST812	1409	0	0	0.00	gsl-1.5/statistics/kurtosis.c
813	ST813	1347	0	0	0.00	gsl-1.5/statistics/lag1.c
814	ST814	1414	0	0	0.00	gsl-1.5/statistics/p_variance.c
815	ST815	1389	0	0	0.00	gsl-1.5/statistics/minmax.c
816	ST816	1377	0	0	0.00	gsl-1.5/statistics/ttest.c

817	ST817	1370	0	0	0.00	gsl-1.5/statistics/median.c
818	ST818	1432	0	0	0.00	gsl-1.5/statistics/covariance.c
819	ST819	1403	0	0	0.00	gsl-1.5/statistics/quantiles.c
820	ST820	421	0	0	0.00	gsl-1.5/statistics/wmean.c
821	ST821	450	0	0	0.00	gsl-1.5/statistics/wvariance.c
822	ST822	444	0	0	0.00	gsl-1.5/statistics/wabsdev.c
823	ST823	439	0	0	0.00	gsl-1.5/statistics/wskew.c
824	ST824	450	0	0	0.00	gsl-1.5/statistics/wkurtosis.c
825	ST825	2984	19	24	0.79	gsl-1.5/statistics/test.c
826	ST826	26440	21	417	0.05	gsl-1.5/statistics/test_nist.c
827	IN827	2835	32	54	0.59	gsl-1.5/integration/qpsrt.c
828	IN828	1967	21	46	0.46	gsl-1.5/integration/qpsrt2.c
829	IN829	5683	38	146	0.26	gsl-1.5/integration/qelg.c
830	IN830	3093	18	90	0.20	gsl-1.5/integration/qc25c.c
831	IN831	5820	18	163	0.11	gsl-1.5/integration/qc25s.c
832	IN832	4302	20	110	0.18	gsl-1.5/integration/qc25f.c
833	IN833	1559	18	33	0.55	gsl-1.5/integration/ptsort.c
834	IN834	3653	21	91	0.23	gsl-1.5/integration/util.c
835	IN835	1613	18	27	0.67	gsl-1.5/integration/err.c
836	IN836	354	2	8	0.25	gsl-1.5/integration/positivity.c
837	IN837	1218	18	13	1.38	gsl-1.5/integration/append.c
838	IN838	1256	18	16	1.13	gsl-1.5/integration/initialise.c
839	IN839	358	0	11	0.00	gsl-1.5/integration/set_initial.c
840	IN840	211	0	8	0.00	gsl-1.5/integration/reset.c
841	IN841	8990	31	166	0.19	gsl-1.5/integration/gsl_integration.h
842	IN842	2445	26	37	0.70	gsl-1.5/integration/qk15.c
843	IN843	2756	26	44	0.59	gsl-1.5/integration/qk21.c
844	IN844	3230	26	57	0.46	gsl-1.5/integration/qk31.c
845	IN845	3731	26	69	0.38	gsl-1.5/integration/qk41.c
846	IN846	4298	27	82	0.33	gsl-1.5/integration/qk51.c
847	IN847	4705	26	94	0.28	gsl-1.5/integration/qk61.c
848	IN848	3179	20	61	0.33	gsl-1.5/integration/qk.c
849	IN849	5338	29	121	0.24	gsl-1.5/integration/qng.c
850	IN850	6914	33	147	0.22	gsl-1.5/integration/qng.h
851	IN851	6900	26	182	0.14	gsl-1.5/integration/qag.c
852	IN852	13637	60	379	0.16	gsl-1.5/integration/qags.c
853	IN853	11803	51	338	0.15	gsl-1.5/integration/qagp.c
854	IN854	3821	38	89	0.43	gsl-1.5/integration/workspace.c
855	IN855	5872	22	162	0.14	gsl-1.5/integration/qcheb.c
856	IN856	5566	26	143	0.18	gsl-1.5/integration/qawc.c
857	IN857	4324	18	131	0.14	gsl-1.5/integration/qmomo.c
858	IN858	5764	27	141	0.19	gsl-1.5/integration/qaws.c
859	IN859	8944	38	267	0.14	gsl-1.5/integration/qmomof.c
860	IN860	10795	42	310	0.14	gsl-1.5/integration/qawo.c
861	IN861	6270	23	188	0.12	gsl-1.5/integration/qawf.c
862	IN862	83938	64	1601	0.04	gsl-1.5/integration/test.c
863	IN863	5799	63	109	0.58	gsl-1.5/integration/tests.c
864	IN864	1749	18	25	0.72	gsl-1.5/integration/tests.h
865	IP865	2426	43	6	7.17	gsl-1.5/interpolation/bsearch.h
866	IP866	5882	28	87	0.32	gsl-1.5/interpolation/gsl_interp.h
867	IP867	2635	20	46	0.43	gsl-1.5/interpolation/gsl_spline.h
868	IP868	1905	20	46	0.43	gsl-1.5/interpolation/accel.c
869	IP869	9374	34	289	0.12	gsl-1.5/interpolation/akima.c
870	IP870	1245	20	19	1.05	gsl-1.5/interpolation/bsearch.c
871	IP871	12255	38	360	0.11	gsl-1.5/interpolation/cspline.c
872	IP872	5794	20	174	0.11	gsl-1.5/interpolation/interp.c
873	IP873	4537	26	148	0.18	gsl-1.5/interpolation/linear.c

874	IP874	1445	21	13	1.62	gsl-1.5/interpolation/integ_eval.h
875	IP875	4898	18	130	0.14	gsl-1.5/interpolation/spline.c
876	IP876	4264	18	115	0.16	gsl-1.5/interpolation/poly.c
877	IP877	7070	27	172	0.16	gsl-1.5/interpolation/test.c
878	HI878	963	18	6	3.00	gsl-1.5/histogram/urand.c
879	HI879	1907	22	44	0.50	gsl-1.5/histogram/find.c
880	HI880	1362	18	23	0.78	gsl-1.5/histogram/find2d.c
881	HI881	4113	19	66	0.29	gsl-1.5/histogram/gsl_histogram.h
882	HI882	5615	19	99	0.19	gsl-1.5/histogram/gsl_histogram2d.h
883	HI883	1421	18	24	0.75	gsl-1.5/histogram/add.c
884	HI884	1713	18	35	0.51	gsl-1.5/histogram/get.c
885	HI885	3811	25	124	0.20	gsl-1.5/histogram/init.c
886	HI886	1121	18	16	1.13	gsl-1.5/histogram/params.c
887	HI887	1016	18	10	1.80	gsl-1.5/histogram/reset.c
888	HI888	2961	18	78	0.23	gsl-1.5/histogram/file.c
889	HI889	3476	23	97	0.24	gsl-1.5/histogram/pdf.c
890	HI890	1725	18	31	0.58	gsl-1.5/histogram/add2d.c
891	HI891	2488	18	60	0.30	gsl-1.5/histogram/get2d.c
892	HI892	6808	28	211	0.13	gsl-1.5/histogram/init2d.c
893	HI893	1410	18	32	0.56	gsl-1.5/histogram/params2d.c
894	HI894	1059	18	11	1.64	gsl-1.5/histogram/reset2d.c
895	HI895	4489	18	121	0.15	gsl-1.5/histogram/file2d.c
896	HI896	4401	24	124	0.19	gsl-1.5/histogram/pdf2d.c
897	HI897	2778	36	50	0.72	gsl-1.5/histogram/calloc_range.c
898	HI898	3807	42	81	0.52	gsl-1.5/histogram/calloc_range2d.c
899	HI899	2148	37	38	0.97	gsl-1.5/histogram/copy.c
900	HI900	2365	37	44	0.84	gsl-1.5/histogram/copy2d.c
901	HI901	2429	33	61	0.54	gsl-1.5/histogram/maxval.c
902	HI902	3328	45	78	0.58	gsl-1.5/histogram/maxval2d.c
903	HI903	3908	65	95	0.68	gsl-1.5/histogram/oper.c
904	HI904	4269	65	103	0.63	gsl-1.5/histogram/oper2d.c
905	HI905	3172	47	64	0.73	gsl-1.5/histogram/stat.c
906	HI906	5817	70	153	0.46	gsl-1.5/histogram/stat2d.c
907	HI907	1241	18	17	1.06	gsl-1.5/histogram/test.c
908	HI908	10410	19	366	0.05	gsl-1.5/histogram/test1d.c
909	HI909	18260	23	607	0.04	gsl-1.5/histogram/test2d.c
910	HI910	2587	18	56	0.32	gsl-1.5/histogram/test1d_resample.c
911	HI911	3248	18	72	0.25	gsl-1.5/histogram/test2d_resample.c
912	HI912	4189	18	75	0.24	gsl-1.5/histogram/test1d_trap.c
913	HI913	7196	18	125	0.14	gsl-1.5/histogram/test2d_trap.c
914	OD914	139	0	0	0.00	gsl-1.5/ode-initval/odeiv_util.h
915	OD915	7671	98	83	1.18	gsl-1.5/ode-initval/gsl_odeiv.h
916	OD916	2213	20	44	0.45	gsl-1.5/ode-initval/control.c
917	OD917	4450	24	122	0.20	gsl-1.5/ode-initval/cstd.c
918	OD918	4692	24	129	0.19	gsl-1.5/ode-initval/cscal.c
919	OD919	4707	30	137	0.22	gsl-1.5/ode-initval/evolve.c
920	OD920	2114	21	52	0.40	gsl-1.5/ode-initval/step.c
921	OD921	4494	29	138	0.21	gsl-1.5/ode-initval/rk2.c
922	OD922	3821	28	109	0.26	gsl-1.5/ode-initval/rk2imp.c
923	OD923	4536	33	137	0.24	gsl-1.5/ode-initval/rk4.c
924	OD924	5058	28	147	0.19	gsl-1.5/ode-initval/rk4imp.c
925	OD925	7841	31	236	0.13	gsl-1.5/ode-initval/rkf45.c
926	OD926	10866	41	344	0.12	gsl-1.5/ode-initval/rk8pd.c
927	OD927	8015	37	238	0.16	gsl-1.5/ode-initval/rkck.c
928	OD928	12871	67	346	0.19	gsl-1.5/ode-initval/bsimp.c
929	OD929	3501	28	97	0.29	gsl-1.5/ode-initval/gear1.c
930	OD930	5547	48	132	0.36	gsl-1.5/ode-initval/gear2.c

931	OD931	16796	29	528	0.05	gsl-1.5/ode-initval/test.c
932	RO932	1232	22	0	0.00	gsl-1.5/roots/roots.h
933	RO933	3701	19	70	0.27	gsl-1.5/roots/gsl_roots.h
934	RO934	3430	21	73	0.29	gsl-1.5/roots/bisection.c
935	RO935	4890	25	163	0.15	gsl-1.5/roots/brent.c
936	RO936	4630	36	97	0.37	gsl-1.5/roots/falsepos.c
937	RO937	2561	26	46	0.57	gsl-1.5/roots/newton.c
938	RO938	2872	32	50	0.64	gsl-1.5/roots/secant.c
939	RO939	3501	35	72	0.49	gsl-1.5/roots/steffenson.c
940	RO940	2321	18	46	0.39	gsl-1.5/roots/convergence.c
941	RO941	2594	20	66	0.30	gsl-1.5/roots/fsolver.c
942	RO942	2115	19	48	0.40	gsl-1.5/roots/fdfsolver.c
943	RO943	9881	25	214	0.12	gsl-1.5/roots/test.c
944	RO944	4045	38	153	0.25	gsl-1.5/roots/test_funcs.c
945	RO945	3037	18	76	0.24	gsl-1.5/roots/test.h
946	MR946	1040	18	10	1.80	gsl-1.5/multiroots/enorm.c
947	MR947	9551	19	288	0.07	gsl-1.5/multiroots/dogleg.c
948	MR948	6081	22	99	0.22	gsl-1.5/multiroots/gsl_multiroots.h
949	MR949	2466	19	57	0.33	gsl-1.5/multiroots/fdjac.c
950	MR950	3882	20	110	0.18	gsl-1.5/multiroots/fsolver.c
951	MR951	4227	20	121	0.17	gsl-1.5/multiroots/fdfsolver.c
952	MR952	2010	18	51	0.35	gsl-1.5/multiroots/convergence.c
953	MR953	3679	20	85	0.24	gsl-1.5/multiroots/newton.c
954	MR954	5082	22	133	0.17	gsl-1.5/multiroots/gnewton.c
955	MR955	4387	25	110	0.23	gsl-1.5/multiroots/dnewton.c
956	MR956	9906	38	292	0.13	gsl-1.5/multiroots/broyden.c
957	MR957	14775	40	465	0.09	gsl-1.5/multiroots/hybrid.c
958	MR958	14114	40	422	0.09	gsl-1.5/multiroots/hybridj.c
959	MR959	7019	20	144	0.14	gsl-1.5/multiroots/test.c
960	MR960	16212	55	525	0.10	gsl-1.5/multiroots/test_funcs.c
961	MR961	3602	18	46	0.39	gsl-1.5/multiroots/test_funcs.h
962	CD962	4625	35	88	0.40	gsl-1.5/cdf/beta_inc.c
963	CD963	364	0	19	0.00	gsl-1.5/cdf/rat_eval.h
964	CD964	123912	0	1413	0.00	gsl-1.5/cdf/test_auto.c
965	CD965	5792	20	70	0.29	gsl-1.5/cdf/gsl_cdf.h
966	CD966	1327	18	22	0.82	gsl-1.5/cdf/beta.c
967	CD967	1283	18	30	0.60	gsl-1.5/cdf/cauchy.c
968	CD968	1489	18	44	0.41	gsl-1.5/cdf/cauchyinv.c
969	CD969	1067	18	10	1.80	gsl-1.5/cdf/chisq.c
970	CD970	1078	18	10	1.80	gsl-1.5/cdf/chisqinv.c
971	CD971	1318	23	26	0.88	gsl-1.5/cdf/exponential.c
972	CD972	1086	18	12	1.50	gsl-1.5/cdf/exponentialinv.c
973	CD973	1711	24	34	0.71	gsl-1.5/cdf/fdist.c
974	CD974	1299	18	36	0.50	gsl-1.5/cdf/flat.c
975	CD975	1260	18	30	0.60	gsl-1.5/cdf/flatinv.c
976	CD976	2815	37	28	1.32	gsl-1.5/cdf/gamma.c
977	CD977	3962	34	110	0.31	gsl-1.5/cdf/gammainv.c
978	CD978	7156	59	231	0.26	gsl-1.5/cdf/gauss.c
979	CD979	4130	26	133	0.20	gsl-1.5/cdf/gaussinv.c
980	CD980	1240	18	22	0.82	gsl-1.5/cdf/gumbel1.c
981	CD981	1354	18	30	0.60	gsl-1.5/cdf/gumbel1inv.c
982	CD982	1287	18	30	0.60	gsl-1.5/cdf/gumbel2.c
983	CD983	1346	18	30	0.60	gsl-1.5/cdf/gumbel2inv.c
984	CD984	1274	18	30	0.60	gsl-1.5/cdf/laplace.c
985	CD985	1461	18	44	0.41	gsl-1.5/cdf/laplaceinv.c
986	CD986	1294	18	30	0.60	gsl-1.5/cdf/logistic.c
987	CD987	1314	18	30	0.60	gsl-1.5/cdf/logisticinv.c

988	CD988	1208	18	14	1.29	gsl-1.5/cdf/lognormal.c
989	CD989	1437	18	32	0.56	gsl-1.5/cdf/lognormalinv.c
990	CD990	1224	18	28	0.64	gsl-1.5/cdf/pareto.c
991	CD991	1331	18	30	0.60	gsl-1.5/cdf/paretoinv.c
992	CD992	1118	18	14	1.29	gsl-1.5/cdf/rayleigh.c
993	CD993	1345	18	30	0.60	gsl-1.5/cdf/rayleighinv.c
994	CD994	5718	41	149	0.28	gsl-1.5/cdf/tdist.c
995	CD995	4882	28	149	0.19	gsl-1.5/cdf/tdistinv.c
996	CD996	1103	18	12	1.50	gsl-1.5/cdf/weibull.c
997	CD997	1342	18	30	0.60	gsl-1.5/cdf/weibullinv.c
998	CD998	54029	32	749	0.04	gsl-1.5/cdf/test.c

- Data Set 4: Six categories of data mining programs which are Association Rules (AR), Support Vector Machine (SV), Genetic Algorithms (GA), Fuzzy Logic (FL), Neural Network (NN), and Decision Tree (DT).

NO.	Code	Size (Bytes)	Lines of Comment	Lines of Code	Ratio Comment Per Code	Filename
1	AR001	5205	41	103	0.40	bodon/source/Apriori.cpp
2	AR002	4244	76	16	4.75	bodon/source/Apriori.hpp
3	AR003	7923	47	183	0.26	bodon/source/Apriori_Trie.cpp
4	AR004	3392	44	32	1.38	bodon/source/Apriori_Trie.hpp
5	AR005	4884	29	114	0.25	bodon/source/Input_Output_Manager.cpp
6	AR006	2512	37	21	1.76	bodon/source/Input_Output_Manager.hpp
7	AR007	4075	35	94	0.37	bodon/source/Trie.cpp
8	AR008	2366	34	29	1.17	bodon/source/Trie.hpp
9	AR009	795	18	1	18.00	bodon/source/common.hpp
10	AR010	5437	31	112	0.28	bodon/source/main.cpp
11	AR011	38508	408	538	0.76	borgelt/apriori/src/apriori.c
12	AR012	34166	399	503	0.79	borgelt/apriori/src/tract.c
13	AR013	9108	85	81	1.05	borgelt/apriori/src/tract.h
14	AR014	70828	880	995	0.88	borgelt/apriori/src/istree.c
15	AR015	7978	97	61	1.59	borgelt/apriori/src/istree.h
16	AR016	15712	209	156	1.34	borgelt/util/src/vecops.c
17	AR017	1579	17	12	1.42	borgelt/util/src/vecops.h
18	AR018	4076	62	58	1.07	borgelt/util/src/params.c
19	AR019	720	9	4	2.25	borgelt/util/src/params.h
20	AR020	10908	139	150	0.93	borgelt/util/src/tfscan.c
21	AR021	4115	53	30	1.77	borgelt/util/src/tfscan.h
22	AR022	21022	265	263	1.01	borgelt/util/src/symtab.c
23	AR023	5230	53	33	1.61	borgelt/util/src/symtab.h
24	AR024	38525	112	81	1.38	borgelt/util/src/scan.c
25	AR025	4862	28	2	14.00	borgelt/util/src/scan.h
26	AR026	4166	60	57	1.05	borgelt/util/src/listops.c
27	AR027	1066	16	7	2.29	borgelt/util/src/listops.h
28	AR028	13403	138	161	0.86	borgelt/util/src/nstats.c
29	AR029	2907	26	30	0.87	borgelt/util/src/nstats.h
30	AR030	5319	55	51	1.08	borgelt/util/src/parse.c
31	AR031	3353	46	2	23.00	borgelt/util/src/parse.h

32	AR032	29752	353	428	0.82	borgelt/eclat/src/bitmat.c
33	AR033	3796	38	26	1.46	borgelt/eclat/src/bitmat.h
34	AR034	19005	214	258	0.83	borgelt/eclat/src/eclat.c
35	AR035	8638	9	308	0.03	goethals/apriori/AprioriSets.cpp
36	AR036	1791	8	43	0.19	goethals/apriori/AprioriSets.h
37	AR037	3029	13	129	0.10	goethals/apriori/Data.cpp
38	AR038	720	6	25	0.24	goethals/apriori/Data.h
39	AR039	641	6	20	0.30	goethals/apriori/Item.cpp
40	AR040	833	6	19	0.32	goethals/apriori/Item.h
41	AR041	1608	9	26	0.35	goethals/apriori/aprioritest.cpp
42	AR042	4767	6	166	0.04	goethals/dic/DIC.cpp
43	AR043	1047	6	28	0.21	goethals/dic/DIC.h
44	AR044	3013	13	121	0.11	goethals/dic/Data.cpp
45	AR045	739	6	25	0.24	goethals/dic/Data.h
46	AR046	649	6	17	0.35	goethals/dic/Item.cpp
47	AR047	1309	10	25	0.40	goethals/dic/Item.h
48	AR048	1377	6	25	0.24	goethals/dic/dictest.cpp
49	AR049	2954	12	127	0.09	goethals/eclat/data.cpp
50	AR050	716	6	25	0.24	goethals/eclat/data.h
51	AR051	6209	19	180	0.11	goethals/eclat/eclat.cpp
52	AR052	1116	6	27	0.22	goethals/eclat/eclat.h
53	AR053	549	6	16	0.38	goethals/eclat/item.h
54	AR054	1453	6	25	0.24	goethals/eclat/testeclat.cpp
55	AR055	2994	13	117	0.11	goethals/fpgrowth/data.cpp
56	AR056	735	6	25	0.24	goethals/fpgrowth/data.h
57	AR057	2255	7	64	0.11	goethals/fpgrowth/fpgrowth.cpp
58	AR058	563	6	15	0.40	goethals/fpgrowth/fpgrowth.h
59	AR059	5072	6	171	0.04	goethals/fpgrowth/fptree.cpp
60	AR060	1131	6	32	0.19	goethals/fpgrowth/fptree.h
61	AR061	1091	6	42	0.14	goethals/fpgrowth/item.cpp
62	AR062	1102	6	33	0.18	goethals/fpgrowth/item.h
63	AR063	1541	6	26	0.23	goethals/fpgrowth/testfpgrowth.cpp
64	AR064	4463	11	135	0.08	goethals/rules/AprioriRules.cpp
65	AR065	868	0	36	0.00	goethals/rules/AprioriRules.h
66	AR066	641	6	20	0.30	goethals/rules/Item.cpp
67	AR067	833	6	19	0.32	goethals/rules/Item.h
68	AR068	1245	7	24	0.29	goethals/rules/ruletest.cpp
69	SV001	4810	48	68	0.71	SvmFu-3.1/src/lib/SvmFuSvmBase.h
70	SV002	9945	28	274	0.10	SvmFu-3.1/src/lib/SvmFuSvmBase.cpp
71	SV003	4150	54	46	1.17	SvmFu-3.1/src/lib/SvmFuSvmKernCache.h
72	SV004	16080	131	362	0.36	SvmFu-3.1/src/lib/SvmFuSvmKernCache.cpp
73	SV005	7521	70	106	0.66	SvmFu-3.1/src/lib/SvmFuSvmLargeOpt.h
74	SV006	29678	152	711	0.21	SvmFu-3.1/src/lib/SvmFuSvmLargeOpt.cpp
75	SV007	5699	71	49	1.45	SvmFu-3.1/src/lib/SvmFuSvmSmallOpt.h
76	SV008	12514	35	338	0.10	SvmFu-3.1/src/lib/SvmFuSvmSmallOpt.cpp
77	SV009	1930	19	23	0.83	SvmFu-3.1/src/lib/SvmFuSvmTest.h
78	SV010	2820	28	48	0.58	SvmFu-3.1/src/lib/SvmFuSvmTest.cpp
79	SV011	1408	18	10	1.80	SvmFu-3.1/src/lib/SvmFuSvmTypedefs.h
80	SV012	2130	27	10	2.70	SvmFu-3.1/src/lib/SvmFuSvmConstants.h
81	SV013	3071	42	0	0.00	SvmFu-3.1/src/lib/SvmFuSvmTypes.h
82	SV014	2069	26	10	2.60	SvmFu-3.1/src/lib/SvmFuSvmConstants.cpp
83	SV015	1357	24	7	3.43	SvmFu-3.1/src/lib/SvmFuSvmDataPoint.h
84	SV016	21874	248	312	0.79	SvmFu-3.1/src/lib/getopt.c
85	SV017	4273	57	17	3.35	SvmFu-3.1/src/lib/getopt.h
86	SV018	4324	30	21	1.43	SvmFu-3.1/src/lib/getopt1.c
87	SV019	11182	38	299	0.13	SvmFu-3.1/src/clients/svmfutrain.cpp
88	SV020	10212	46	267	0.17	SvmFu-3.1/src/clients/svmfutest.cpp

89	SV021	14595	54	406	0.13	SvmFu-3.1/src/clients/svmfutrainmulti.cpp
90	SV022	12299	86	331	0.26	SvmFu-3.1/src/clients/clientincludes.h
91	SV023	3933	40	29	1.38	SvmFu-3.1/src/clients/kernelfuncs.h
92	SV024	9731	40	215	0.19	SvmFu-3.1/src/clients/kernelfuncs.cpp
93	SV025	3829	2	153	0.01	libsvm-2.6/svm-predict.c
94	SV026	5569	11	217	0.05	libsvm-2.6/svm-scale.c
95	SV027	6740	9	262	0.03	libsvm-2.6/svm-train.c
96	SV028	55358	186	2320	0.08	libsvm-2.6/svm.cpp
97	SV029	2146	17	48	0.35	libsvm-2.6/svm.h
98	SV030	12756	174	152	1.14	svm_light/svm_common.h
99	SV031	1999	32	4	8.00	svm_light/kernel.h
100	SV032	8580	18	141	0.13	svm_light/svm_learn.h
101	SV033	135887	499	3459	0.14	svm_light/svm_learn.c
102	SV034	17693	35	336	0.10	svm_light/svm_learn_main.c
103	SV035	7102	37	146	0.25	svm_light/svm_classify.c
104	SV036	25471	78	816	0.10	svm_light/svm_common.c
105	SV037	7124	41	153	0.27	svm_light/svm_loqo.c
106	SV038	29667	162	871	0.19	svm_light/svm_hideo.c
107	SV039	2124	18	42	0.43	SVMTorch/Cache.h
108	SV040	5157	34	161	0.21	SVMTorch/Cache.cc
109	SV041	3866	16	99	0.16	SVMTorch/convert.cc
110	SV042	2857	16	66	0.24	SVMTorch/Kernel.h
111	SV043	1526	16	8	2.00	SVMTorch/OldIOTorch.h
112	SV044	1973	20	8	2.50	SVMTorch/general.h
113	SV045	2565	18	56	0.32	SVMTorch/SVM.h
114	SV046	5149	20	181	0.11	SVMTorch/OldIOTorch.cc
115	SV047	6064	25	245	0.10	SVMTorch/Kernel.cc
116	SV048	11733	22	419	0.05	SVMTorch/StandardSVM.cc
117	SV049	7684	23	268	0.09	SVMTorch/SVM.cc
118	SV050	1625	16	16	1.00	SVMTorch/UserKernel.h
119	SV051	1990	16	27	0.59	SVMTorch/StandardSVM.h
120	SV052	2427	16	55	0.29	SVMTorch/UserKernel.cc
121	SV053	12570	29	388	0.07	SVMTorch/SVMTest.cc
122	SV054	1493	16	10	1.60	SVMTorch/general.cc
123	SV055	5933	17	240	0.07	SVMTorch/IOTorch.cc
124	SV056	1774	16	10	1.60	SVMTorch/IOTorch.h
125	SV057	14295	20	431	0.05	SVMTorch/SVMTorch.cc
126	GA001	35558	157	791	0.20	galib245/ga/GA1DArrayGenome.C
127	GA002	6789	52	96	0.54	galib245/ga/GA1DArrayGenome.h
128	GA003	22695	109	488	0.22	galib245/ga/GA1DBinStrGenome.C
129	GA004	5815	42	91	0.46	galib245/ga/GA1DBinStrGenome.h
130	GA005	22276	32	604	0.05	galib245/ga/GA2DArrayGenome.C
131	GA006	4745	18	94	0.19	galib245/ga/GA2DArrayGenome.h
132	GA007	21452	43	599	0.07	galib245/ga/GA2DBinStrGenome.C
133	GA008	4095	16	75	0.21	galib245/ga/GA2DBinStrGenome.h
134	GA009	32205	45	885	0.05	galib245/ga/GA3DArrayGenome.C
135	GA010	5228	17	104	0.16	galib245/ga/GA3DArrayGenome.h
136	GA011	31493	52	906	0.06	galib245/ga/GA3DBinStrGenome.C
137	GA012	4633	15	86	0.17	galib245/ga/GA3DBinStrGenome.h
138	GA013	13646	144	282	0.51	galib245/ga/GAAllele.C
139	GA014	8166	62	108	0.57	galib245/ga/GAAllele.h
140	GA015	3185	29	60	0.48	galib245/ga/GAArray.h
141	GA016	17921	42	411	0.10	galib245/ga/GABaseGA.C
142	GA017	10159	41	136	0.30	galib245/ga/GABaseGA.h
143	GA018	8994	49	194	0.25	galib245/ga/GABin2DecGenome.C
144	GA019	5480	41	84	0.49	galib245/ga/GABin2DecGenome.h
145	GA020	2036	23	27	0.85	galib245/ga/GABinStr.C

146	GA021	2622	23	51	0.45	galib245/ga/GABinStr.h
147	GA022	2146	11	66	0.17	galib245/ga/GADCrowdingGA.C
148	GA023	855	8	9	0.89	galib245/ga/GADCrowdingGA.h
149	GA024	12173	30	360	0.08	galib245/ga/GADemeGA.C
150	GA025	4531	25	66	0.38	galib245/ga/GADemeGA.h
151	GA026	915	12	10	1.20	galib245/ga/GAEvalData.h
152	GA027	2536	19	57	0.33	galib245/ga/GAGenome.C
153	GA028	13488	176	88	2.00	galib245/ga/GAGenome.h
154	GA029	6722	21	201	0.10	galib245/ga/GAIncGA.C
155	GA030	3705	35	41	0.85	galib245/ga/GAIncGA.h
156	GA031	5928	50	96	0.52	galib245/ga/GAList.C
157	GA032	8004	83	95	0.87	galib245/ga/GAList.h
158	GA033	8342	87	158	0.55	galib245/ga/GAListBASE.C
159	GA034	7298	107	46	2.33	galib245/ga/GAListBASE.h
160	GA035	18055	123	401	0.31	galib245/ga/GAListGenome.C
161	GA036	2656	16	34	0.47	galib245/ga/GAListGenome.h
162	GA037	1368	7	28	0.25	galib245/ga/GAMask.h
163	GA038	3725	43	31	1.39	galib245/ga/GANode.h
164	GA039	16640	94	467	0.20	galib245/ga/GAParameter.C
165	GA040	3800	20	66	0.30	galib245/ga/GAParameter.h
166	GA041	24384	168	529	0.32	galib245/ga/GAPopulation.C
167	GA042	10013	44	242	0.18	galib245/ga/GASStateGA.C
168	GA043	9300	74	139	0.53	galib245/ga/GAPopulation.h
169	GA044	9873	40	217	0.18	galib245/ga/GARealGenome.C
170	GA045	2631	11	48	0.23	galib245/ga/GARealGenome.h
171	GA046	2708	15	44	0.34	galib245/ga/GASStateGA.h
172	GA047	9711	21	9	2.33	galib245/ga/GAScaling.C
173	GA048	10935	101	26	3.88	galib245/ga/GAScaling.h
174	GA049	15494	58	0	0.00	galib245/ga/GASelector.C
175	GA050	10720	69	21	3.29	galib245/ga/GASelector.h
176	GA051	6729	39	168	0.23	galib245/ga/GASimpleGA.C
177	GA052	2435	11	41	0.27	galib245/ga/GASimpleGA.h
178	GA053	18352	77	456	0.17	galib245/ga/GAStatistics.C
179	GA054	6928	49	144	0.34	galib245/ga/GAStatistics.h
180	GA055	3181	18	60	0.30	galib245/ga/GAStringGenome.C
181	GA056	2369	9	44	0.20	galib245/ga/GAStringGenome.h
182	GA057	10875	99	156	0.63	galib245/ga/GATree.C
183	GA058	9484	103	110	0.94	galib245/ga/GATree.h
184	GA059	21784	164	481	0.34	galib245/ga/GATreeBASE.C
185	GA060	9595	131	69	1.90	galib245/ga/GATreeBASE.h
186	GA061	9842	65	222	0.29	galib245/ga/GATreeGenome.C
187	GA062	2891	20	33	0.61	galib245/ga/GATreeGenome.h
188	GA063	8722	153	0	0.00	galib245/ga/ga.h
189	GA064	10491	91	142	0.64	galib245/ga/gabincvt.C
190	GA065	1766	20	8	2.50	galib245/ga/gabincvt.h
191	GA066	16348	147	1	147.00	galib245/ga/gaconfig.h
192	GA067	5196	10	128	0.08	galib245/ga/gaerror.C
193	GA068	3592	40	61	0.66	galib245/ga/gaerror.h
194	GA069	3912	52	23	2.26	galib245/ga/gaid.h
195	GA070	8271	52	69	0.75	galib245/ga/garandom.C
196	GA071	6288	58	12	4.83	galib245/ga/garandom.h
197	GA072	954	11	11	1.00	galib245/ga/gatypes.h
198	GA073	490	8	1	8.00	galib245/ga/gaversion.h
199	GA074	3194	47	95	0.49	ECGA/cache.cpp
200	GA075	1128	19	20	0.95	ECGA/cache.hpp
201	GA076	2509	32	63	0.51	ECGA/chromosome.cpp
202	GA077	1085	13	22	0.59	ECGA/chromosome.hpp

203	GA078	2627	40	52	0.77	ECGA/ecga.cpp
204	GA079	497	10	10	1.00	ECGA/ecga.hpp
205	GA080	932	17	28	0.61	ECGA/gene.cpp
206	GA081	960	11	20	0.55	ECGA/gene.hpp
207	GA082	4072	35	161	0.22	ECGA/intlist.cpp
208	GA083	1400	15	36	0.42	ECGA/intlist.hpp
209	GA084	3863	24	81	0.30	ECGA/main.cpp
210	GA085	6238	59	166	0.36	ECGA/mpm.cpp
211	GA086	1383	16	27	0.59	ECGA/mpm.hpp
212	GA087	1691	29	34	0.85	ECGA/objfunc.cpp
213	GA088	414	11	2	5.50	ECGA/objfunc.hpp
214	GA089	1062	21	18	1.17	ECGA/parameter.hpp
215	GA090	5347	46	138	0.33	ECGA/population.cpp
216	GA091	1749	24	32	0.75	ECGA/population.hpp
217	GA092	2791	43	75	0.57	ECGA/random.cpp
218	GA093	1258	18	25	0.72	ECGA/random.hpp
219	GA094	3458	52	100	0.52	ECGA/subset.cpp
220	GA095	1353	16	27	0.59	ECGA/subset.hpp
221	GA096	2542	26	100	0.26	ECGA/utility.cpp
222	GA097	696	10	11	0.91	ECGA/utility.hpp
223	GA098	9540	127	247	0.51	LLGA/chromosome.cpp
224	GA099	1662	20	30	0.67	LLGA/chromosome.hpp
225	GA100	1065	15	24	0.63	LLGA/gene.cpp
226	GA101	1137	13	27	0.48	LLGA/gene.hpp
227	GA102	1115	20	31	0.65	LLGA/geneArray.cpp
228	GA103	801	13	14	0.93	LLGA/geneArray.hpp
229	GA104	3979	58	65	0.89	LLGA/llga.cpp
230	GA105	2558	41	35	1.17	LLGA/llga.hpp
231	GA106	12391	131	266	0.49	LLGA/llga_io.cpp
232	GA107	1851	35	27	1.30	LLGA/llga_io.hpp
233	GA108	2783	49	45	1.09	LLGA/objfunc.cpp
234	GA109	595	11	3	3.67	LLGA/objfunc.hpp
235	GA110	8563	78	226	0.35	LLGA/population.cpp
236	GA111	2718	29	48	0.60	LLGA/population.hpp
237	GA112	2190	33	69	0.48	LLGA/random.c
238	GA113	642	14	4	3.50	LLGA/random.h
239	GA114	1186	13	33	0.39	LLGA/util.cpp
240	GA115	586	10	7	1.43	LLGA/util.hpp
241	FL001	2331	60	24	2.50	ffll_src_2_2_1/COGDefuzz.cpp
242	FL002	7293	179	89	2.01	ffll_src_2_2_1/COGDefuzzSetObj.cpp
243	FL003	2183	37	24	1.54	ffll_src_2_2_1/COGDefuzzSetObj.h
244	FL004	4911	126	50	2.52	ffll_src_2_2_1/COGDefuzzVarObj.cpp
245	FL005	3730	87	13	6.69	ffll_src_2_2_1/COGDefuzzVarObj.h
246	FL006	1735	59	10	5.90	ffll_src_2_2_1/DefuzzSetObj.cpp
247	FL007	1405	24	11	2.18	ffll_src_2_2_1/DefuzzSetObj.h
248	FL008	1823	61	10	6.10	ffll_src_2_2_1/DefuzzVarObj.cpp
249	FL009	1563	29	14	2.07	ffll_src_2_2_1/DefuzzVarObj.h
250	FL010	11520	287	134	2.14	ffll_src_2_2_1/FFLLAPI.cpp
251	FL011	1759	22	11	2.00	ffll_src_2_2_1/FFLLAPI.h
252	FL012	11374	240	220	1.09	ffll_src_2_2_1/FFLLBase.cpp
253	FL013	8165	92	67	1.37	ffll_src_2_2_1/FFLLBase.h
254	FL014	64582	1097	1021	1.07	ffll_src_2_2_1/FuzzyModelBase.cpp
255	FL015	6091	55	76	0.72	ffll_src_2_2_1/FuzzyModelBase.h
256	FL016	5814	149	75	1.99	ffll_src_2_2_1/FuzzyOutSet.cpp
257	FL017	2296	36	16	2.25	ffll_src_2_2_1/FuzzyOutSet.h
258	FL018	9241	210	143	1.47	ffll_src_2_2_1/FuzzyOutVariable.cpp
259	FL019	4033	57	29	1.97	ffll_src_2_2_1/FuzzyOutVariable.h

260	FL020	482	6	6	1.00	ffll_src_2_2_1/FuzzyOutVariableBase.cpp
261	FL021	1071	18	6	3.00	ffll_src_2_2_1/FuzzyOutVariableBase.h
262	FL022	13159	248	256	0.97	ffll_src_2_2_1/FuzzySetBase.cpp
263	FL023	5362	72	56	1.29	ffll_src_2_2_1/FuzzySetBase.h
264	FL024	33655	732	537	1.36	ffll_src_2_2_1/FuzzyVariableBase.cpp
265	FL025	5799	48	69	0.70	ffll_src_2_2_1/FuzzyVariableBase.h
266	FL026	4518	107	57	1.88	ffll_src_2_2_1/MOMDefuzzSetObj.cpp
267	FL027	1680	31	14	2.21	ffll_src_2_2_1/MOMDefuzzSetObj.h
268	FL028	4385	119	51	2.33	ffll_src_2_2_1/MOMDefuzzVarObj.cpp
269	FL029	1988	32	13	2.46	ffll_src_2_2_1/MOMDefuzzVarObj.h
270	FL030	11998	217	212	1.02	ffll_src_2_2_1/MemberFuncBase.cpp
271	FL031	3829	45	46	0.98	ffll_src_2_2_1/MemberFuncBase.h
272	FL032	16995	393	200	1.97	ffll_src_2_2_1/MemberFuncSCurve.cpp
273	FL033	2098	28	21	1.33	ffll_src_2_2_1/MemberFuncSCurve.h
274	FL034	5595	170	55	3.09	ffll_src_2_2_1/MemberFuncSingle.cpp
275	FL035	2118	30	18	1.67	ffll_src_2_2_1/MemberFuncSingle.h
276	FL036	10305	249	150	1.66	ffll_src_2_2_1/MemberFuncTrap.cpp
277	FL037	1773	26	17	1.53	ffll_src_2_2_1/MemberFuncTrap.h
278	FL038	8700	234	120	1.95	ffll_src_2_2_1/MemberFuncTri.cpp
279	FL039	1784	26	17	1.53	ffll_src_2_2_1/MemberFuncTri.h
280	FL040	5080	136	89	1.53	ffll_src_2_2_1/RuleArray.cpp
281	FL041	2005	31	22	1.41	ffll_src_2_2_1/RuleArray.h
282	FL042	571	22	0	0.00	ffll_src_2_2_1/debug.h
283	NN001	2348	16	76	0.21	nnlib-1.0/layer.cpp
284	NN002	729	8	20	0.40	nnlib-1.0/layer.h
285	NN003	1536	3	49	0.06	nnlib-1.0/main.cpp
286	NN004	3355	14	95	0.15	nnlib-1.0/neuralnet.cpp
287	NN005	702	4	19	0.21	nnlib-1.0/neuralnet.h
288	NN006	2659	17	98	0.17	nnlib-1.0/neuron.cpp
289	NN007	1105	12	31	0.39	nnlib-1.0/neuron.h
290	NN008	402	3	8	0.38	nnlib-1.0/activation.h
291	NN009	838	7	24	0.29	nnlib-1.0/activation.cpp
292	NN010	7576	104	127	0.82	NNLib-0.1/graph.h
293	NN011	2067	36	40	0.90	NNLib-0.1/graph.cpp
294	NN012	7458	93	101	0.92	NNLib-0.1/nn_base.h
295	NN013	6069	58	143	0.41	NNLib-0.1/nn_base.cpp
296	NN014	7578	89	114	0.78	NNLib-0.1/struct.h
297	NN015	7964	52	192	0.27	NNLib-0.1/struct.cpp
298	NN016	980	18	11	1.64	NNLib-0.1/mlp.h
299	NN017	3574	28	89	0.31	NNLib-0.1/mlp.cpp
300	NN018	997	19	12	1.58	NNLib-0.1/rmlp.h
301	NN019	3658	22	71	0.31	NNLib-0.1/rmlp.cpp
302	NN020	1484	0	46	0.00	NNLib-0.1/elems.cpp
303	NN021	2383	0	59	0.00	NNLib-0.1/prova.cpp
304	NN022	2502	1	70	0.01	NNLib-0.1/prova.rmlp.cpp
305	NN023	9851	141	71	1.99	inanna-0.3.3/src/trainer.h
306	NN024	13074	94	249	0.38	inanna-0.3.3/src/dataformats.cc
307	NN025	10638	137	83	1.65	inanna-0.3.3/src/equalization.h
308	NN026	6121	114	30	3.80	inanna-0.3.3/src/annetwork.h
309	NN027	3807	62	16	3.88	inanna-0.3.3/src/dataformat.h
310	NN028	3108	20	62	0.32	inanna-0.3.3/src/topology.cc
311	NN029	11703	67	279	0.24	inanna-0.3.3/src/patternset.cc
312	NN030	13835	123	234	0.53	inanna-0.3.3/src/trainer.cc
313	NN031	8258	133	81	1.64	inanna-0.3.3/src/termination.h
314	NN032	4570	72	23	3.13	inanna-0.3.3/src/annfilef.h
315	NN033	6600	108	34	3.18	inanna-0.3.3/src/topology.h
316	NN034	10400	79	139	0.57	inanna-0.3.3/src/termination.cc

317	NN035	7151	111	60	1.85	inanna-0.3.3/src/freenet.h
318	NN036	676	0	11	0.00	inanna-0.3.3/src/Makefile.am
319	NN037	9951	0	251	0.00	inanna-0.3.3/src/Makefile.in
320	NN038	15403	92	347	0.27	inanna-0.3.3/src/freenet.cc
321	NN039	13621	100	262	0.38	inanna-0.3.3/src/equalization.cc
322	NN040	1675	7	34	0.21	inanna-0.3.3/src/connection.cc
323	NN041	12087	182	77	2.36	inanna-0.3.3/src/patternset.h
324	NN042	7116	56	143	0.39	inanna-0.3.3/src/neuron.cc
325	NN043	11445	179	87	2.06	inanna-0.3.3/src/neuron.h
326	NN044	3020	24	54	0.44	inanna-0.3.3/src/annetwork.cc
327	NN045	2019	23	33	0.70	inanna-0.3.3/src/initializer.h
328	NN046	3146	23	47	0.49	inanna-0.3.3/src/dataformat.cc
329	NN047	3213	33	14	2.36	inanna-0.3.3/src/dataformats.h
330	NN048	3180	41	31	1.32	inanna-0.3.3/src/connection.h
331	NN049	10289	63	183	0.34	inanna-0.3.3/src/annfilef.cc
332	DT001	1623	13	53	0.25	R8/Src/average.c
333	DT002	10178	81	242	0.33	R8/Src/besttree.c
334	DT003	11977	99	324	0.31	R8/Src/build.c
335	DT004	888	14	11	1.27	R8/Src/buildex.i
336	DT005	3875	14	134	0.10	R8/Src/c4.5.c
337	DT006	3266	27	92	0.29	R8/Src/c4.5rules.c
338	DT007	3691	27	105	0.26	R8/Src/classify.c
339	DT008	1124	9	34	0.26	R8/Src/confmat.c
340	DT009	10106	79	291	0.27	R8/Src/consult.c
341	DT010	6409	55	205	0.27	R8/Src/consultr.c
342	DT011	5791	39	140	0.28	R8/Src/contin.c
343	DT012	1213	8	1	8.00	R8/Src/defns.i
344	DT013	3813	38	79	0.48	R8/Src/discr.c
345	DT014	1767	37	23	1.61	R8/Src/extern.i
346	DT015	1107	13	16	0.81	R8/Src/genlogs.c
347	DT016	5445	49	128	0.38	R8/Src/genrules.c
348	DT017	3862	35	111	0.32	R8/Src/getdata.c
349	DT018	7693	67	179	0.37	R8/Src/getnames.c
350	DT019	985	10	24	0.42	R8/Src/getopt.c
351	DT020	671	7	13	0.54	R8/Src/header.c
352	DT021	5319	66	106	0.62	R8/Src/info.c
353	DT022	4428	40	114	0.35	R8/Src/makerules.c
354	DT023	6351	32	202	0.16	R8/Src/prune.c
355	DT024	11340	108	304	0.36	R8/Src/prunerule.c
356	DT025	10741	74	321	0.23	R8/Src/rules.c
357	DT026	1107	21	14	1.50	R8/Src/rulex.i
358	DT027	18803	155	582	0.27	R8/Src/siftrules.c
359	DT028	1470	17	34	0.50	R8/Src/sort.c
360	DT029	5156	36	125	0.29	R8/Src/st-thresh.c
361	DT030	1842	18	39	0.46	R8/Src/stats.c
362	DT031	9713	65	265	0.25	R8/Src/subset.c
363	DT032	6690	33	224	0.15	R8/Src/testrules.c
364	DT033	14086	125	394	0.32	R8/Src/trees.c
365	DT034	3058	48	64	0.75	R8/Src/types.i
366	DT035	6388	61	205	0.30	R8/Src/userint.c
367	DT036	3383	27	92	0.29	R8/Src/xval-prep.c
368	DT037	2442	4	94	0.04	cn2/att_order.c
369	DT038	4327	20	150	0.13	cn2/attribute.c
370	DT039	7625	11	290	0.04	cn2/ckrl_gen.c
371	DT040	6363	70	160	0.44	cn2/ckrl_rules.c
372	DT041	9799	27	295	0.09	cn2/cn.c
373	DT042	5264	31	153	0.20	cn2/cn_externs.h

374	DT043	9180	67	149	0.45	cn2/cn_header.h
375	DT044	10661	6	350	0.02	cn2/cn_print_thing.c
376	DT045	5003	22	163	0.13	cn2/create.c
377	DT046	414	5	12	0.42	cn2/debug.c
378	DT047	2082	12	63	0.19	cn2/example.c
379	DT048	10602	19	354	0.05	cn2/execute.c
380	DT049	5121	25	145	0.17	cn2/filter.c
381	DT050	4432	49	114	0.43	cn2/heap.c
382	DT051	301	5	0	0.00	cn2/heap.h
383	DT052	587	6	10	0.60	cn2/input.h
384	DT053	8376	5	288	0.02	cn2/interact.c
385	DT054	11179	17	470	0.04	cn2/interact_utils.c
386	DT055	30569	16	1327	0.01	cn2/lex.yy.c
387	DT056	971	4	45	0.09	cn2/list.c
388	DT057	1582	6	44	0.14	cn2/main.c
389	DT058	428	8	0	0.00	cn2/mdep.h
390	DT059	300	7	0	0.00	cn2/mlt_float.h
391	DT060	3634	26	101	0.26	cn2/names.c
392	DT061	7086	24	218	0.11	cn2/peccles.c
393	DT062	4066	9	151	0.06	cn2/print_gen_thing.c
394	DT063	908	6	8	0.75	cn2/qlalloc.h
395	DT064	9382	49	103	0.48	cn2/quickfit.c
396	DT065	1912	32	28	1.14	cn2/reserved.h
397	DT066	1144	10	35	0.29	cn2/robin.c
398	DT067	2295	5	70	0.07	cn2/rule_reader.c
399	DT068	19936	77	493	0.16	cn2/specialise.c
400	DT069	766	4	24	0.17	cn2/test.c
401	DT070	14918	5	499	0.01	cn2/trace.c
402	DT071	44299	163	788	0.21	cn2/y.tab.c
403	DT072	6425	56	101	0.55	lmdt/code/discard.c
404	DT073	11784	92	247	0.37	lmdt/code/encode.c
405	DT074	9917	85	170	0.50	lmdt/code/er.c
406	DT075	15789	145	283	0.51	lmdt/code/lm.c
407	DT076	9714	79	195	0.41	lmdt/code/load.c
408	DT077	13653	101	255	0.40	lmdt/code/print.c
409	DT078	21476	170	403	0.42	lmdt/code/prune.c
410	DT079	12481	110	195	0.56	lmdt/code/test.c
411	DT080	17489	126	337	0.37	lmdt/code/tree.c
412	DT081	4059	49	43	1.14	lmdt/code/utils.c
413	DT082	6803	77	107	0.72	lmdt/code/lm.h

APPENDIX C

SAMPLE PAGES OF A FEATURE VECTOR FILE

Below is an excerpt of the first twelve lines from the feature vector file of Data Set 1. The first line is the number of features or keywords. The second line starts with #n and is followed by a list of keywords. The ten remaining lines are for representing the ten software components. Each line of a software component consists of the tfxidf values (see Section 3.2.1) of the corresponding keywords listed in the second line. Due to the limitation of the paper width, the actual line (several hundred characters long) cannot be presented in one physical line. Therefore, line numbers are added at the end of each line as superscripts to indicate the actual line numbers.

542[1]

#n charact plancomp global printf randomtablestyp fig symbol class sequenc place report ident header reason reqcomp pattern special stop maketransit edit pass mean grant recurs check form exit arcstyp random previou begin date xdstring failur stringcomp data match queue smith argv rang chang reclaimmemori tail push cpp storag expect goal point parent declar construct rousso unimpl execut initi zeroth part transit pars findmin good argc send similar txt stdout index list put consum addit max atoi plan lisp throw work stream return search row make map hold level intcel getel word char argument dynam children logic constcomp lookup empti provid abnorm endif exampl concept acknowledg occur attempt function univers rotat destructor root increment destin support text decis duplic ptr differ event deriv enum memori read listitr platform time oper bound maximum capac buffer written lower open machin organ figur loop updat retriev ismark attach correct enter simpl scienc bind brown represent error rule distribut inform istream paramet advanc skip eventcompar fals makeempti close boolean thu permiss final main attribut full kei requir fee remov field unit deep lbh case april compar konstantino sizeof long stderr box entri owncalloc gener jun getlabel cout repres jul low acct gap strncpy operatorcomp address brusd extern bunch perform add space depend valid arc tue val ifstream pertain mphf logicnod storedvalu basic isempti ispastend elementat dim postal doesn content free subtre abstract product doubl consist currents problem endl purpos length dsexcept proven public iostream arctyp overflow modif hash present string smallest impli rotatewithleftchild iter posit minim creat item size strlen pointer march infin state exercis base fclose fail request malloc fprintf clear sourc algorithm randomint valu link specif verticestyp prior math resiz success independ line recent topandpop nullnod crule static order redistribut combin arrai friend fox ifdef conflict num setlabel adjac sun cerr sum pat fopen find test sortedqueu note contain total visit dimens ifndef pop reach upper modul effect tabl print start code nest info front output underflow min messag continu back modifi applic destroi itr satisfi union copyright str

clone routin incorpor need stdlib switch input understand defin default surasmith driver notfound fput precondit knr vertic method warranti stdio typedef process append child intersect fstream access term number mark program standard made findmax unmark stack constrain commerci set comparison addmemb singl norm implement regist virtual track advertis ostream strcmp conflictcomp node initialvalu file delet suitabl neg high member multipl exist elementtyp split document minitem structur copi boundari build privat result maintain assum step vector constructor planningst heurist printtre determin run command displai pair element compil mutat forward seed isful oop bit signal top subtract group alloc linkedlist call vertexarrai handl binari alphabet break degre struct factdatabas sort prioriti unsign deletemin constant type equal getlin label select electron store intern actual slbagiter assign counter depart name express stdc left end largest hashtabl object merg true depth termin weight vertex count ret explicit current flag req chen simul comput fact rantab asgn indic constraint produc heap usag const wed tmp null linkcomp comp comment notifi softwar warn email formula mystr unmarktransit notif head adjust notic variabl graph write found bindcompar librarian take filenam condit correctli entir bool book wartik person fix void remain version fit rotatewithrightchild integ cell factcompar insert slbag temp wether templat theelement stepcomp path tree[2]
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.609438 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3.465736 0 0 0
0 0.6931472 0 0 0 4.8520303 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 1.3862944 0 0 0 0
0 0 0 0 0 0.6931472 0 5.5451775 0.6931472 0 0 2.7725887 1.3862944 0 0 0 0 0 0 0 0 0 2.944439 2.7725887 1.3862944
0 0 0.6931472 0 0 1.3862944 0 0 0 0 0 0 0 0 1.609438 0 0 0 0.6931472 0 0 0 0 0 0 0 3.295837 0 1.3862944 0 0 0
0.6931472 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0.6931472 0 0 0 0 0 0.0 0 0 0
0.0 0 0 0 0.6931472 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0.6931472 0 0.6931472
0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472
0 0 0 0 0 3.465736 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 1.609438 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.3862944 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 1.3862944 0 0.6931472 0 0 0 0 0 0 0 2.7725887 0 0 0.0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0.6931472 0
0.6931472 0 0 0.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 4.158883
0 0 0 0 0 1.3862944 0 0 2.0794415 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
AI001[3]
0 0 0 0 0 0 0 0.0 0 0 0 0 0 3.583519 0 0 0 0 0 0 0 5.6664267 0 1.3862944 0 0 0 0 0 0 0 0 1.3862944 0 0 0 43.944492 0 0 0
0 0 6.802395 0 32.38883 0 0 0 0 0 0 0 0 0 1.609438 0 0 5.3752785 0 0 0 0 0 0 0 2.0794415 0 0 0 0 40.202538 0 0 0 0 0
0 0 0 0 0 0.0 4.828314 0 0.0 0 0 0 0 25.644932 0 0 0 0 0 0 0 0 4.158883 3.465736 0 0.6931472 0 0 0.6931472 0 0
1.0986123 4.8520303 0 0 0 0 0 0.6931472 0 0 0 0 0 0 3.5263605 0 0 0 0 0 0 2.0794415 0 0 0 0 0.6931472 0 0 0
1.3862944 0 0 32.4966 0 0 0 0 0 0 0.6931472 0 5.5451775 0.6931472 0 0 2.7725887 1.3862944 0 0 0 0 0 1.609438 0 0
0 2.944439 2.7725887 1.3862944 0 0 0.6931472 0 0 1.3862944 4.158883 0 0 0 0 0 0 23.070858 1.609438 0 0 0
0.6931472 0 0 0 1.609438 0 0 0 0 0 1.3862944 0 0 0 0.6931472 0.6931472 0 0 0 0 0 3.4011974 0 0 0 0 0 0.6931472 0 0
0 0 0 0 0 0 0.6931472 0 0 0 0 0 0.6931472 0 0 0 0 0 0.0 0 0 0 0 0.0 0 0 0 0 0.6931472 0 0 0 0 0.6931472 0 0 0 0 1.9459101
0 0 0 0 0 0 42.810024 0 2.6390574 0 0 0.6931472 0 0 0 0 0 0 0 0 0.6931472 0 0.6931472 0 0 0 0 0 0 0 0 0 0 1.9459101
0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 45.842686 0 1.0986123 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 36.635616 0 0
0.6931472 0 0 0 0 0 3.465736 0 0 0 0.6931472 0 0 0 0 0 2.0794415 4.7957907 0 0 0 0 0 1.609438 0 0 0.6931472 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 39.578793 0 0.6931472 0 0 0 45.064262 0 1.3862944 4.394449
0.6931472 0 0 8.671115 0 0 0 0 2.7725887 0 0 0.0 0 0 0 0 0 0 0 0 0 4.6051702 0 0 0 5.278115 0 0 6.591674 0 24.953299
0 0 0 0 0 0 0 0 0 0 0 0 0 1.7917595 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.3862944 0 24.849066 0 0 3.0910425 0
0.6931472 0 0.6931472 0 0 0.0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0
3.5263605 0 0.6931472 4.158883 0 0 0 0 0 1.3862944 54.41916 0 2.0794415 4.394449 3.4011974 0 0 0 0.6931472 0 0
0 0 0 0 0 0 0.6931472 0 0.0 0 0 0 0 0 0 0 0 10.986123 0 0 12.17809 0 0 0 0 0 AI002[4]
0 0 0 0 0 0 2.5649493 0 0 3.1780539 0 0 0 0 0 0 0 0 0 0 5.6664267 0 1.3862944 0 3.583519 0 0 0 0 0 0 1.3862944 0 0 0
2.1972246 0 0 0 0 0 3.4011974 0 0 0 0 0 0 0 0 0 0 0 0 1.609438 0 0 1.7917595 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0.0 0 0 0 0.0 0 0 0 0 0 0 0 3.295837 0 0 0 0 0 0 0 3.465736 0 0 0 0 0.6931472 0 3.8066626 10.986123 4.8520303
0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 1.3862944 0 0 29.788551 0 0 0 0 0 0 0.6931472
0 5.5451775 0.6931472 0 0 2.7725887 1.3862944 0 12.542316 0 0 0 0 0 0 0 2.944439 2.7725887 2.7725887 0 0
0.6931472 0 0 1.3862944 6.931472 0 0 0 0 0 0 0 1.609438 0 0 0 0.6931472 0 0 0 1.609438 0 0 0 0 0 1.3862944 0 0 0
0.6931472 0.6931472 0 0 0 5.8377304 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0.6931472 0 0 0 0 0
0.0 0 0 0 0.0 0 0 0 0.6931472 0 0 1.3862944 0 0.6931472 0 0 0 0 1.9459101 0 0 3.1780539 0 0 0 60.323215 0 0 0 0
0.6931472 0 0 0 0 0 0 0 0 0.6931472 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
38.789967 0 0 0 0 0 0 13.621371 0 0 1.3862944 0 0 0 0 0 0 0 7.327123 0 0 0.6931472 0 0 0 0 0 3.465736 0 0 0
0.6931472 0 0 0 0 0 0 2.3978953 0 0 0 0 0 1.609438 0 0 0.6931472 0 0 3.0445225 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.3862944 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 32.18876 0 1.3862944 1.0986123 0.6931472 0 0 0 0 0 0 0 2.7725887 0 0
0.0 0 0 0 2.3025851 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7.167038 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
23.070858 0 1.3862944 0 0 0 0 0 0 0.6931472 0 0.6931472 0 0 0.0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.828314 0 0 0 0 0.6931472 0
0 1.3862944 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 4.158883 0 0 0 0 0 1.3862944 0 0 2.0794415 0 6.802395 0 0 0
0.6931472 3.4011974 0 0 0 3.0910425 0 0 0 0.6931472 0 0.0 0 0 0 0 0 0 0 0 2.1972246 0 0 6.089045 0 0 0 0 0 AI003[5]
0 0 0 0 0 0 2.5649493 0.0 0 0 0 0 3.583519 0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 1.3862944 0 0 0 8.788898 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.609438 0 0 0 0 0 0 0 0 0 0 0.6931472 3.5263605 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.0 0 0 0 0.0 0 0 0 0

0 0 0 2.1972246 0 0 0 0 0 0 3.465736 0 0.6931472 0 0 0.6931472 0 0 0 4.8520303 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0
2.944439 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 1.3862944 0 0 5.4161005 0 0 0 0 0 0 0.6931472 0 5.5451775 0.6931472 0
0 2.7725887 1.3862944 0 0 0 0 0 0 0 0 2.944439 2.7725887 1.3862944 0 0 0.6931472 0 0 1.3862944 0 0 0 0 0 0 0
1.0986123 1.609438 0 0 0 0.6931472 0 0 0 1.609438 0 0 2.8332133 0 0 1.3862944 0 0 0 0.6931472 0.6931472 0 0 0 0
0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0.6931472 0 0 0 0 0 0.0 0 0 0 0 0.0 0 0 0 0 0.6931472 0 0
1.3862944 0 0.6931472 0 0 0 0 0 0 0 0 0 0 21.405012 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0.6931472 0 0.6931472 0 0 0
0 0 0 0 0 0 0 0 0.6931472 0 0 1.7917595 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7.052721 0 1.0986123 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0.6931472 0 0 0 0 0 3.465736 0 0 0 0.6931472 0 0 0 0 0 0.6931472 0 0 0 0 0 0 1.609438 0 0 0.6931472 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 9.133567 0 0.6931472 0 2.7080503 0 11.266066 0 1.3862944 0
0.6931472 0 0 0 0 0 0 0 2.7725887 0 0 0.0 0 0 1.609438 0 0 0 0 0 0 0 0 0 0 0 0 2.1972246 0 2.7725887 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.9459101 0 13.183348 0 1.3862944 0 0 0 0 0 0 0.6931472 0 0.6931472 0 0 0.0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 4.158883 0 0 0 0 0 1.3862944 0
0 2.0794415 0 0 0 0 0 0.6931472 0 0 0 0 0 0 3.5263605 0 0.6931472 0 0.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 AI004[6]
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.609438 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 2.8332133 0 0.0 0 0 0 0.0 0 0 0 0 0 0 0 1.0986123 0 0 0 0 0
0 4.158883 3.465736 0 0 0 0 0.6931472 0 0 0 4.8520303 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.6931472 0 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0.6931472 0 5.5451775 0.6931472 0 0 2.7725887 1.3862944 0 0 0 0 0 0 0
0 0 2.944439 2.7725887 1.3862944 2.1972246 0 0.6931472 0 0 1.3862944 0 0 0 0 0 0 0 0 1.609438 0 0 0 0.6931472 0
0 0 1.609438 0 1.0986123 0 0 0 1.3862944 0 0 0 0.6931472 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0
0.6931472 0 0 0 0 0 0.6931472 0 0 0 0 1.0986123 0.0 0 0 0 0 0.0 0.6931472 0 0 0.6931472 0 0 0 0 0.6931472 0 0 0 0 0 0
0 0 0 0 0 5.8377304 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0.6931472 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 6.591674 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 3.465736 0 0 0
0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 1.609438 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 0
0.6931472 0 0 0 0 0 1.3862944 0 0.6931472 0 0 0 0 0 0 0 2.7725887 0 0 0.0 0 0 0 13.815511 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0.6931472 0 0.6931472 0 0 0.0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 4.158883 0 0 0 0 0 1.3862944 0 0
2.0794415 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 AI005[7]
0 0 0 0 0 0 0 0 0 0 0 0 3.583519 0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1.609438 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0.0 0 0 0 0 0 0 0 0 0 0 0 0 0 2.7725887
3.465736 0 0.6931472 0 0 0.6931472 0 0 0 4.8520303 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472
0 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0.6931472 0 5.5451775 0.6931472 0 0 2.7725887 1.3862944 0 0 0 0 0 0 0 0 0
2.944439 2.7725887 1.3862944 3.295837 0 0.6931472 0 0 1.3862944 0 0 0 0 0 0 0 0 1.609438 0 0 0 0.6931472 0 0 0 0
0 0 0 3.295837 0 1.3862944 0 0 0 0.6931472 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0.6931472 0 0 0
0 0 0.6931472 0 0 0 0 0 0.0 0 0 0 0 0.0 0 0 0 0 0.6931472 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0
0 0 0 0 0 0.6931472 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.0986123 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 3.465736 0 0 0 0.6931472 0 0 0 0 0 0.6931472 0 0 0 0 0 0
1.609438 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 1.3862944 0
0.6931472 0 0 0 0 0 0 0 2.7725887 0 0 0.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0.6931472 0 0.6931472 0 0 0.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0
1.3862944 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 4.158883 0 0 0 0 0 1.3862944 0 0 2.0794415 0 0 0 0 0 0.6931472 0 0 0 0 0
0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 AI006[8]
0 0 0 0 0 0 12.824747 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1.609438 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0 0.0 1.609438 0 0.0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 3.465736 0 0 0 0 0.6931472 0 0 2.1972246 4.8520303 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0
0.6931472 10.579082 0 0 0 0.6931472 0 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0.6931472 133.23318 5.5451775 0.6931472 0
0 2.7725887 1.3862944 0 3.583519 0 0 0 1.609438 0 0 0 2.944439 2.7725887 1.3862944 0 0 0.6931472 0 0 1.3862944
0 0 0 0 0 0 0 1.0986123 1.609438 0 0 0 0.6931472 0 0 0 1.609438 0 2.1972246 0 0 0 1.3862944 0 0 0 0.6931472
0.6931472 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 38.789967 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0.6931472 0 0 0 0 2.1972246
0.0 0 0 0 0 0.0 0 0 0 0 0.6931472 0 0 0 0 0.6931472 0 0 0 0 1.9459101 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0
0.6931472 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3.8918202 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 3.465736 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 1.609438 0 0 0.6931472 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 1.3862944 2.1972246 0.6931472 0 0 0
0 0 0 0 2.7725887 0 2.6390574 0.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2.1972246 0 1.3862944 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 5.8377304 0 0 0 1.3862944 0 0 0 0 0 0 0.6931472 0 0.6931472 0 0 0.0 0 0 0 0 1.3862944 0 0 0 0 0
0 0 0 0 0 0 0 0.6931472 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 4.158883 0 0 0 0 0 1.3862944 0 0 2.0794415
39.55004 0 0 0 15.22665 0.6931472 0 0 0 0 0 0 0 0 0.6931472 0 0.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 AI007[9]
0 0 0 0 0 0 0 0 0.0 0 0 0 0 3.583519 0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1.609438 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.0 0 0 0 0.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3.465736 0 0.6931472 0 0 0.6931472 0 0 0 4.8520303 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0
0.6931472 0 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0.6931472 49.486614 5.5451775 0.6931472 0 0 2.7725887 1.3862944 0 0
0 0 0 0 0 0 0 2.944439 2.7725887 1.3862944 0 0 0.6931472 0 0 1.3862944 0 0 0 0 0 0 0 1.0986123 1.609438 0 0 0
0.6931472 0 0 0 1.609438 0 0 0 0 0 1.3862944 0 0 0 0.6931472 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 24.684525

0 0 0 0 0 0.6931472 0 0 0 0 0 0.6931472 0 0 0 0 0 0.0 0 0 0 0 0.0 0 0 0 0 0.6931472 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0.6931472 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 1.7917595 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1.7917595 0 0 0 0 1.0986123 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 3.465736 0 0 0
0.6931472 0 0 0 0 0 0.6931472 0 0 0 0 0 0 1.609438 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0
0 0 0 0 0 0.6931472 0 0 0 0 0 1.3862944 0 0.6931472 0 0 0 0 0 0 2.7725887 0 0 0.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 3.6635616 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3.6635616 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0.6931472 0
0.6931472 0 0 0.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 1.3862944 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 4.158883
0 0 0 0 0 1.3862944 0 0 2.0794415 8.788898 0 0 0 3.8066626 0.6931472 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 AI008[10]
0 0 0 0 0 0 0 0.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 22.839975 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4.394449 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 21.158163 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0.0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1.0986123 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.609438 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 4.394449 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3.8918202 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7.613325 0 0 0 0 0 0 0 0 0 0 0
0.0 0 0 0 0 0 0 0 0 0 0 0.0 0 0 0 0 AI009[11]
0 0 0 0 0 0 56.428886 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.3862944 0 0 0 0 0 0 0 0 1.3862944 2.7080503 0 0 2.1972246 0
41.446533 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.609438 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.0
11.266066 0 0.0 0 0 3.1780539 0 0 0 7.690286 0 0 6.089045 19.183163 0 0 0 3.465736 0 0 0 0 0.6931472 0 0 7.690286
4.8520303 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 14.654246 0.6931472 0 0 0 1.3862944 0 0 0 0
0 0 0 0 0 0.6931472 0 5.5451775 0.6931472 0 0 2.7725887 1.3862944 0 10.750557 0 0 0 1.609438 0 0 0 2.944439
2.7725887 1.3862944 0 0 0.6931472 0 0 1.3862944 2.7725887 0 0 0 10.203592 6.591674 0 4.394449 1.609438 0 0 0
0.6931472 0 0 0 1.609438 0 7.690286 2.8332133 0 0 1.3862944 0 3.0445225 0 0.6931472 0.6931472 0 0 0 0 0 0 0 0 0
7.327123 0 0.6931472 0 208.05527 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0.6931472 0 0 0 0 2.1972246 0.0 5.278115 0 0 0.0
0 0 0 0.6931472 0 0 6.931472 0 0.6931472 0 0 0 0 5.8377304 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0.6931472
0 0.6931472 0 0 2.3025851 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1.3862944 2.1972246 0 0 0 0
3.5263605 0 0 0 0 0 0 11.675461 0 0 5.5451775 0 0 0 0 0 0 0 0 0 0 0.6931472 0 0 0 0 0 3.465736 0 0 0 0.6931472 0 0
2.944439 0 0 0.6931472 2.3978953 0 0 0 0 0 1.609438 0 0 0.6931472 0 0 0 0 69.31472 0 0 0 0 0 43.162113 0 0 0 0
3.4011974 0 0 1.3862944 0 0 0 2.944439 0 0 0 0 0 0.6931472 0 0 0 61.15864 0 1.3862944 2.1972246 0.6931472 0 0 0
0 2.944439 0 0 2.7725887 0 2.6390574 0.0 0 0 0 0 0 0 0 0 2.3025851 0 0 0 0 0 0 6.591674 0 0 0 0 0 0 0 0 0 0 2.944439
0 29.308493 0 0 0 0 0 0 0 8.788898 0 0 0 0 0 0 0 10.9906845 31.191624 7.7836404 0 0 0 1.3862944 0 0 0 0 0 0
0.6931472 0 0.6931472 0 0 0.0 0 0 1.609438 0 1.3862944 0 0 0 0 0 7.613325 0 33.798195 0 0 0 0 0.6931472 0 0
1.3862944 0 0 0 0 0 0 0 0 0 0 0 0 0.6931472 4.158883 0 0 0 0 0 1.3862944 0 0 2.0794415 10.986123 0 0 1.3862944 0
0.6931472 0 0 0 0 0 0 0 0 0.6931472 0 0.0 0 0 0 0 0 0 0 4.394449 0 11.090355 0 0 0 0 0 0 AI010[12]

APPENDIX D

AN EXAMPLE OF A SOURCE CODE ITEMSETS FILE

An example of a source code itemsets file of Data Set 1 (consisting of data structure, information retrieval, and artificial intelligence components) is given below. The first column is the running number used to count the number of software components in a data set. The second column is the code assigned to a software component. The third column is an itemset which contains a list of all include file that a software component contains.

NO.	Code	Include Filename
1	DS001	AATree.h iostream.h
2	DS002	AATree.cpp dsexceptions.h iostream.h
3	DS003	AvlTree.h iostream.h
4	DS004	AvlTree.cpp dsexceptions.h iostream.h
5	DS005	BinaryHeap.h
6	DS006	BinaryHeap.cpp dsexceptions.h vector.h
7	DS007	BinarySearchTree.h iostream.h
8	DS008	BinarySearchTree.cpp dsexceptions.h iostream.h
9	DS009	BinomialQueue.h dsexceptions.h
10	DS010	BinomialQueue.cpp iostream.h vector.h
11	DS011	iostream.h
12	DS012	fstream iostream map sstream string strstrea.h strstream.h vector
13	DS013	AvlTree.h fstream.h iostream.h LinkedList.h mystring.h strstrea.h strstream.h
14	DS014	CursorList.h
15	DS015	CursorList.cpp dsexceptions.h vector.h
16	DS016	DSL.h
17	DS017	dsexceptions.h DSL.cpp iostream.h
18	DS018	DisjSets.h
19	DS019	vector.h
20	DS020	iostream.h
21	DS021	iostream.h
22	DS022	iostream.h
23	DS023	iostream.h

24	DS024	iostream.h
25	DS025	iostream.h mystring.h vector.h
26	DS026	IntCell.h iostream.h
27	DS027	iostream.h
28	DS028	iostream.h mystring.h
29	DS029	iostream.h mystring.h vector.h
30	DS030	iostream.h vector.h
31	DS031	iostream.h
32	DS032	iostream.h
33	DS033	iostream.h matrix.h
34	DS034	iostream.h
35	DS035	iostream.h
36	DS036	iostream.h vector.h
37	DS037	iostream.h limits.h matrix.h
38	DS038	iostream.h matrix.h
39	DS039	iostream.h Random.h
40	DS040	algorithm iostream vector
41	DS041	dsexceptions.h iostream list
42	DS042	iostream set string
43	DS043	IntCell.h iostream.h mystring.h vector.h
44	DS044	fstream iostream limits.h list map sstream stack string vector
45	DS045	fstream.h iostream.h limits.h LinkedList.h mystring.h QueueAr.h SeparateChaining.h strstrea.h strstream.h
46	DS046	IntCell.h
47	DS047	
48	DS048	iostream.h vector.h
49	DS049	dsexceptions.h LeftistHeap.h
50	DS050	iostream.h LeftistHeap.cpp
51	DS051	LinkedList.h
52	DS052	dsexceptions.h iostream.h LinkedList.cpp
53	DS053	iostream.h vector.h
54	DS054	MemoryCell.h
55	DS055	MemoryCell.cpp
56	DS056	dsexceptions.h PairingHeap.h
57	DS057	iostream.h PairingHeap.cpp
58	DS058	iostream.h vector.h
59	DS059	iostream.h QuadraticProbing.h
60	DS060	mystring.h QuadraticProbing.cpp vector.h
61	DS061	QueueAr.h
62	DS062	dsexceptions.h QueueAr.cpp vector.h
63	DS063	Random.h
64	DS064	
65	DS065	RedBlackTree.h
66	DS066	dsexceptions.h iostream.h RedBlackTree.cpp
67	DS067	iostream.h SeparateChaining.h
68	DS068	LinkedList.h mystring.h SeparateChaining.cpp vector.h
69	DS069	vector.h
70	DS070	iostream.h SplayTree.h
71	DS071	dsexceptions.h iostream.h SplayTree.cpp
72	DS072	StackAr.h
73	DS073	dsexceptions.h StackAr.cpp vector.h
74	DS074	iostream.h StackLi.h
75	DS075	dsexceptions.h iostream.h StackLi.cpp
76	DS076	AATree.h iostream.h
77	DS077	AvlTree.h iostream.h
78	DS078	BinaryHeap.h dsexceptions.h iostream.h
79	DS079	BinarySearchTree.h iostream.h
80	DS080	BinomialQueue.h iostream.h

81	DS081	CursorList.h iostream.h
82	DS082	DSL.h iostream.h
83	DS083	DisjSets.h iostream.h
84	DS084	IntCell.h iostream.h
85	DS085	iostream.h LeftistHeap.h
86	DS086	iostream.h LinkedList.h
87	DS087	iostream.h MemoryCell.h mystring.h
88	DS088	dsexceptions.h iostream.h PairingHeap.h vector.h
89	DS089	iostream.h QuadraticProbing.h
90	DS090	iostream.h QueueAr.h
91	DS091	iostream.h Random.h
92	DS092	iostream.h Random.h
93	DS093	iostream.h RedBlackTree.h
94	DS094	iostream.h SeparateChaining.h
95	DS095	vector.h
96	DS096	iostream.h Random.h Sort.h vector.h
97	DS097	iostream.h SplayTree.h
98	DS098	iostream.h StackAr.h
99	DS099	iostream.h StackLi.h
100	DS100	string.cpp
101	DS101	iostream.h Treap.h
102	DS102	iostream.h Treap.h
103	DS103	dsexceptions.h iostream.h limits.h Random.h Treap.cpp
104	DS104	
105	DS105	
106	DS106	vector.h
107	DS107	iostream.h
108	DS108	mystring.h string.h
109	DS109	vector.h
110	DS110	vector.cpp
111	IR001	bv.h
112	IR002	
113	IR003	bv.h stdio.h
114	IR004	bv.h hash.h stdio.h
115	IR005	hash.h stdio.h
116	IR006	
117	IR007	hash.h stdio.h
118	IR008	rantab.h stdio.h string.h types.h
119	IR009	
120	IR010	
121	IR011	math.h stdio.h string.h support.h types.h
122	IR012	comphfns.h pmrandom.h rantab.h stdio.h types.h
123	IR013	stdio.h support.h types.h vheap.h
124	IR014	pmrandom.h
125	IR015	
126	IR016	pmrandom.h rantab.h types.h
127	IR017	
128	IR018	math.h rantab.h regenphf.h stdio.h string.h types.h
129	IR019	comphfns.h rantab.h regenphf.h stdio.h types.h
130	IR020	
131	IR021	pmrandom.h stdio.h support.h types.h
132	IR022	stdio.h types.h
133	IR023	
134	IR024	const.h
135	IR025	limits.h math.h stack stdio.h support.h types.h vheap.h
136	IR026	
137	IR027	ctype.h stdio.h string.h

138	IR028	
139	IR029	ctype.h stdio.h stem.h
140	IR030	ctype.h malloc.h stdio.h stop.h string.h strlist.h
141	IR031	strlist.h
142	IR032	stdio.h stop.h strlist.h
143	IR033	list malloc.h memory.h stdio.h string string.h strlist.h
144	IR034	list string
145	IR035	string.h
146	IR036	string.h
147	IR037	string.h
148	IR038	string.h
149	IR039	string.h
150	IR040	
151	IR041	
152	IR042	stdio.h string.h
153	IR043	
154	IR044	stdio.h string.h
155	IR045	math.h stdio.h string.h
156	IR046	math.h stdio.h string.h
157	IR047	math.h stdio.h string.h
158	AI001	DataList.H
159	AI002	DataNode.H SortedList.H
160	AI003	DataList.H DataNode.H string.h
161	AI004	string.h
162	AI005	DataList.H DataNode.H iostream.h main.H
163	AI006	
164	AI007	Bind.H LogicNode.H
165	AI008	Compare.H
166	AI009	
167	AI010	LogicNode.H Queue.H string.h
168	AI011	Compare.H Queue.H XDString.H
169	AI012	Bind.H iostream.h LogicNode.H Parser.H Queue.H stdlib.h
170	AI013	Bind.H LogicNode.H Queue.H XDString.H
171	AI014	Bind.H iostream.h LogicNode.H Parser.H
172	AI015	DTree.H Formula.H fstream.h iostream.h Node.H stdlib.h String.H strstream.h
173	AI016	
174	AI017	Formula.H Key.H
175	AI018	iostream.h
176	AI019	iostream.h Key.H stdlib.h String.H
177	AI020	iostream.h String.H
178	AI021	DTree.H Formula.H Key.H Node.H
179	AI022	
180	AI023	String.H
181	AI024	
182	AI025	Chromosome.H math.h stdlib.h
183	AI026	
184	AI027	Chromosome.H iostream.h math.h Population.H stdlib.h time.h
185	AI028	
186	AI029	State.H
187	AI030	
188	AI031	Queue.H Searches.H State.H
189	AI032	Searches.H State.H
190	AI033	Searches.H State.H
191	AI034	Decision.H Dimension.H Example.H fstream.h iostream.h Node.H stdlib.h
192	AI035	iostream.h String.H
193	AI036	Dimension.H Example.H Node.H String.H
194	AI037	iostream.h String.H

195	AI038	Decision.H Dimension.H Example.H
196	AI039	iostream.h String.H
197	AI040	Decision.H Dimension.H Example.H iostream.h math.h Node.H
198	AI041	
199	AI042	String.H
200	AI043	
201	AI044	Function.H
202	AI045	
203	AI046	Function.H iostream.h math.h PDP.H stdlib.h
204	AI047	
205	AI048	math.h PDP.H
206	AI049	math.h PDP.H
207	AI050	Function.H
208	AI051	
209	AI052	fstream.h Function.H iostream.h Perceptron.H stdlib.h time.h
210	AI053	
211	AI054	Boundary.H Concept.H Dimension.H Example.H Version.H
212	AI055	Concept.H iostream.h
213	AI056	Concept.H Dimension.H Example.H String.H Version.H
214	AI057	Dimension.H iostream.h
215	AI058	Dimension.H String.H
216	AI059	iostream.h String.H
217	AI060	Dimension.H Example.H iostream.h stdlib.h Version.H
218	AI061	Dimension.H iostream.h String.H
219	AI062	String.H
220	AI063	
221	AI064	Dimension.H Example.H fstream.h iostream.h stdlib.h Version.H
222	AI065	Boundary.H
223	AI066	CausalRuleDatabase.H List.H SIterator.H
224	AI067	Effects.H FactDatabase.H List.H SortedQueue.H
225	AI068	iostream.h
226	AI069	Effects.H
227	AI070	Compare.H SortedList.H Time.H XDString.H
228	AI071	Effects.H FactDatabase.H
229	AI072	Effects.H SortedList.H
230	AI073	CausalRuleDatabase.H Effects.H FactDatabase.H SortedQueue.H TemporalUpdate.H
231	AI074	
232	AI075	CausalRuleDatabase.H Effects.H FactDatabase.H SortedQueue.H TemporalUpdate.H
233	AI076	Compare.H SortedQueue.H
234	AI077	Time.H
235	AI078	iostream.h
236	AI079	
237	AI080	Conflict.H Link.H Plan.H SLBag.H Step.H
238	AI081	Compare.H SLBag.H
239	AI082	Constrain.H SLBag.H SLBagIterator.H Step.H
240	AI083	Compare.H SLBag.H
241	AI084	Heuristic.H Plan.H
242	AI085	State.H
243	AI086	Conflict.H Link.H Plan.H SLBag.H SLBagIterator.H Step.H XDString.H
244	AI087	Compare.H SLBag.H SLBagIterator.H
245	AI088	Operator.H
246	AI089	Compare.H SLBag.H XDString.H
247	AI090	Conflict.H Constrain.H Link.H Operator.H Plan.H Queue.H Requirement.H SLBag.H SLBagIterator.H Step.H
248	AI091	Compare.H Queue.H SLBag.H State.H
249	AI092	Conflict.H Constrain.H Link.H Operator.H Plan.H Requirement.H SLBag.H SLBagIterator.H Step.H XDString.H
250	AI093	Compare.H SLBag.H SLBagIterator.H
251	AI094	SortedQueue.H

252	AI095	State.H
253	AI096	
254	AI097	Conflict.H Constrain.H Link.H Operator.H Plan.H Requirement.H SLBag.H SLBagIterator.H Step.H
255	AI098	Compare.H SLBag.H
256	AI099	Searches.H SortedQueue.H State.H
257	AI100	Conflict.H Constrain.H Heuristic.H Link.H Operator.H Plan.H Requirement.H Searches.H Step.H XDString.H
258	AI101	
259	AI102	Heuristic.H
260	AI103	PlanningState.H State.H
261	AI104	Operator.H
262	AI105	Compare.H SLBag.H XDString.H
263	AI106	SLSet.H XDString.H
264	AI107	PlanningState.H
265	AI108	Compare.H Operator.H Queue.H SLBag.H State.H XDString.H
266	AI109	SortedQueue.H
267	AI110	State.H
268	AI111	
269	AI112	Heuristic.H PlanningState.H Searches.H StateSearches.H
270	AI113	
271	AI114	Searches.H SortedQueue.H State.H
272	AI115	Queue.H Searches.H State.H
273	AI116	Heuristic.H Operator.H PlanningState.H Searches.H SLBag.H StateSearches.H XDString.H

VITA

Songsri Tangsripairoj

Candidate for the Degree of

Doctor of Philosophy

Thesis: A GROWING HIERARCHICAL SELF-ORGANIZING MAP WITH MINING ASSOCIATION RULES FOR SOFTWARE REPOSITORY ORGANIZATION AND VISUALIZATION

Major Field: Computer Science

Biographical:

Personal Data: Born in Nakhonpathom, Thailand, on November 28, 1973, the daughter of Pairin and Srisuda Tangsripairoj.

Education: Graduated from Satri Withaya School, Bangkok, Thailand in March 1990; received Bachelor of Science Degree in Computer Science with Second Class Honors from Thammasat University, Bangkok, Thailand in March 1994; received Master of Science Degree in Computer Science from Mahidol University, Bangkok, Thailand in October 1996. Completed the requirements for the Doctor of Philosophy Degreee with a major in Computer Science at the Computer Science Department at Oklahoma State University in December 2004.

Professional Experience: Employed by Mahidol University as a computer staff member at the Computing Center, 1994 to 1995, and a lecturer at the Department of Computer Science, Faculty of Science, 1995 to present; employed by Computer Science Department, Oklahoma State University as a teaching assistant, August 2000 to December 2004.